REAL TIME MOTION CONTROL FOR NATURAL HUMAN ROBOT INTERACTION

by

SURESH SAMPATHKUMAR

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2013

Copyright © by Suresh Sampathkumar 2013

All Rights Reserved

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Dan Popa for constantly motivating and encouraging me, and also for his invaluable advice during the course of my Masters studies. I would like to also thank Dr. Frank Lewis and Dr. Michael Manry for taking time to serve in my dissertation committee.

I wish to thank my colleagues Isura Ranatunga, Joe Sanford, Ahsan Habib, Nahum Torres, Monica Beltran, Ishan Chakravorthy, Abhishek Thakurdesai, Corina Bogdan, Sumit Das, Namrata Balakrishnan for their help and support. I am grateful to Dr. David Hanson for giving me opportunity to work at Hanson Robotics. I am also grateful to University of Texas Arlington Research Institute for letting me access their research facilities.

Finally, I would like to express my deep gratitude to my parents who have encouraged and inspired me and sponsored my graduate studies. I am also grateful to my mother and father for their sacrifice, encouragement and patience. I also thank several of my friends who have helped me throughout my career. .

November 25, 2013

ABSTRACT

REAL TIME MOTION CONTROL FOR NATURAL

HUMAN ROBOT INTERACTION

Suresh Sampathkumar, M.S

The University of Texas at Arlington, 2013

Supervising Professor: Dan Popa

In this thesis, we present the research work performed with various robots including Android PKD, Zeno and the Atlas robot. The main focus of this thesis is to develop algorithms for vision-based natural human robot interaction in real-time.

Zeno is a childlike robot that we use for the research related to Autism in children. Children with Autism who lack motor skills tend to shy away from humans. In this project we use the robot Zeno that is capable of facial expressions, arm and body motions to teach motor skills to them. For this purpose we developed a program that plays predetermined gestures like hand wave, tummy rub and fist bump in a particular sequence. This was used in the UNT Health Science Center to record the motion of the Autistic children so that we can compare the motion of the child and the robot.

Android PKD is an avatar of the science fiction writer Philip K. Dick. In this project we developed a novel algorithm to distribute the motion between the neck and the eye servo motors to solve the redundancy involved. The robot was able to track faces by moving its eyes and neck in a realistic humanlike motion. Another project was done to mimic the facial expressions and the 3D rotations of the human head by the PKD. In this the mapping between

the facial feature space and the actuator space is addressed using linear and nonlinear methods. Both of these demos and Zeno were exhibited at the Humanoids 2013 Conference in Atlanta.

An algorithm to map the human arm, torso and leg motion to the robot was developed as a project for DARPA Robotics Challenge and also controlling the robot in real-time. The robot used is called Atlas and it is a virtual robot in a Gazebo simulation environment. The issue of singularities has been addressed while mapping the human motion to the robot.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation for Vision-based Human Robot Interaction

Human Robot Interaction (HRI) is the study of interactions between humans and robots. Human Robot Interaction is a multidisciplinary field with contributions from robotics, human computer interaction, artificial intelligence, computer vision, machine learning, natural language understanding, real-time processing and social sciences.

Human Robot Interaction is the development of robotics systems enabling interaction capabilities more similar to human-human interaction has become a hot research topic. Most of the research in this area is carried with a goal to furnish robotics systems with the function to observe, detect and respond to the modes of interaction that people apply naturally with one another.

A major research area that attracts the research in Human Robot Interaction is in the area of Autism. Autism spectrum disorder (ASD) and autism are both general terms for a group of complex disorders of brain development. These disorders are characterized, in varying degrees, by difficulties in social interaction, verbal and nonverbal communication and repetitive behaviors. They include autistic disorder, Rett syndrome, childhood disintegrative disorder, pervasive developmental disorder-not otherwise specified (PDD-NOS) and Asperger syndrome. With the May 2013 publication of the new DSM-5 diagnostic manual, these autism subtypes will be merged into one umbrella diagnosis of ASD. ASD can be associated with intellectual disability, difficulties in motor coordination and attention and physical health issues such as sleep and gastrointestinal disturbances. Autism appears to have its roots in very early brain development. However, the most obvious signs of autism and symptoms of autism tend to emerge between 2 and 3 years of age. Through experimental observations it has been seen

that children with Autism like interacting with Robots. This is one of the major motivations for conducting research in natural human-robot interaction. It is also important for the robots to be socially acceptable so the robots that we use for research must satisfy this criterion. It is also needed to come up with algorithms leading to natural human robot interaction. Several areas that need algorithm development are face tracking, facial expressions, mimicking the human motions for natural HRI. Face tracking is an important ability for the robot to perform natural conversations with the humans. The challenge lies in developing computer vision algorithms to process the image to position the eyes of the robot and to control the servo motors such that the face tracking by the robot looks realistic and occurs in real-time. Facial expression generation is another important aspect of natural HRI. Facially expressive robots find more acceptance while interacting with humans. Facial expressions synthesis and mimicking is an important part of the research that can enable social interaction with humans.

1.2 Challenges in Natural Human Robot Interaction

Human Robot Interaction can be classified into three types: visual interaction, physical interaction and interaction with verbal dialogue. Physical interaction involves sensing of the environment through contact sensors like force feedback sensors. Dialogue based interaction involves the use of voice feedback together with artificial intelligence to interpret the message. Vision-based interaction involves sensing the environment through cameras (normal and stereo), 3D depth sensors like Microsoft Kinect etc. There are many challenges involved in vision-based Human Robot Interaction. Face tracking by the robot, mimicking the motion of the human by the robot, motion control for distributing motion to actuators that have redundancy, facial expressions synthesis and mapping to the robot are some of the aspects requiring new algorithm development.

Face tracking by the robot is a challenging task as it needs robust algorithms. There is considerable redundancy in the neck and eye actuators in order to track the human face. One of the challenges is how to servo the motors such that the robot motion looks more human like

instead of looking unnatural. The other challenge is to being able to detect a face in a given image in real-time. Numerous face detection algorithms that are available in the open source community are not deterministic. This means we cannot guarantee that a face is detected during fixed intervals of time. This deterministic requirement is important if we implement a control algorithm for real-time face tracking. Another challenge is the computational time needed to detect a face in a given image. Generally for the control algorithms generally we need to actuate the motors at 100 Hz bandwidths to get a smooth motion.

There are also several research challenges in human motion detection and in mapping the human motion to the robot by considering the robot's kinematic constraints. There are software available for tracking human skeleton using 3D depth cameras like Microsoft Kinect. The challenge here is how we use those non real-time algorithms for controlling a real-time robot. Another research challenge in mapping the human kinematics to that of the robot kinematics might involve singularities and there is a need to take care of this as well.

For mapping the facial expressions of the human to that of the robot, the challenge is to be able to track those critical muscle movements that are responsible for a certain facial expressions and to map these points that are being tracked to the actuator space of the robot that create facial expressions.

In order to accomplish real-time vision-based interaction we need to rely on the processing requirements as well. For example, the current face detection using OpenCV takes approximately 500ms to detect a face using a full 640x480 pixel image. To improve this we might carry out processing of these algorithms in FPGA and DSP chips. There is considerable complexity involved in designing a real-time, distributed, embedded systems for this purpose, so the control of the motors is deterministic and communication between the control hardware and the vision acquisition hardware is real-time.

Robotic systems are inherently real-time systems. The robot has to respond within a defined latency for it to perform tasks successfully. For example, the robot that does the

assembly in a manufacturing plant has to pick and place with precise timing requirements. A robot that does the obstacle avoidance has to respond in real-time when it encounters an obstacle. Robots that incorporate vision for manipulation tasks also tend to be real-time, for example the robotic arm that has a fixed stereo camera to estimate the depth of an object for grabbing, need to establish its orientation in real-time so that it can plan its trajectory for grabbing it. Another example is using the 3D depth acquisition devices like Kinect to build the map and localize itself in it also have to be done in real-time.

In order to handle these tasks efficiently, we need choose appropriate Software and Hardware design model to accomplish these tasks. Computer Vision algorithms tend to be computationally expensive. For example, to be able to run the Simultaneous Localization and Mapping (SLAM) on standard desktops we need processors with Graphics Processing Unit (GPU) in it for performing the computationally intensive algorithms. Processors that are good for performance like Intel Core i7 based on x86 or x64 architecture running on Windows/Linux operating systems may be well suited to get increased average throughput but are not efficient in handling tasks that needs to be run periodically with deadline constraints. Context switching in such processors frequently could turn out to be expensive and the deadlines of the real-time control and sensing tasks may fail to meet. So the use of such systems to perform deterministic low level tasks is not a feasible solution.

The solution for handling sensing and actuating tasks that have specific scheduling rate such as 100Hz or 1 KHz is then to use a processor that supports real-time operating system. So for a complete hardware system we can use a high performance x86 or ARM based processor for processing vision related algorithms and use another processor that runs a real-time operating systems that can do the sensing and the actuation in real-time. For communication between the vision system and the control system we can choose a real-time communication protocol like CAN, EtherCAT [1][2].

We need to choose appropriate Software model for the vision and the control system. For processing the vision algorithms from Kinect we can use the open source library OpenNI [3] that provides the driver support for the depth acquisition devices. For image processing applications we can use the OpenCV library. Since all of these libraries can be installed easily in Robot Operating System (ROS) [4] it would be a good choice for building the vision system. Since ROS is heavily supported in Linux, we can use Linux as our operating system for vision system. For the real-time sensing and control of robots we can consider any real time system that can handle real-time tasks. In our case we use real-time controller from National instruments called sbRIO [5] for the purpose of controlling and sensing.

1.3 Description of the Work Conducted

In this thesis, we describe the algorithm development and implementation for natural human robot interaction. We start with reviewing the current literature on algorithms for human robot interaction including image and position based visual servoing, control of humanoid robots and focusses on developing new algorithms. For interaction with humanoid robots like Androids, a novel motion distribution algorithm is developed to distribute the motion between eye servos and neck servos to solve the redundancy in actuation such that the movement appears humanlike. Facial expression imitation by the android is implemented by mapping the feature points of the human face to that of the actuator space of the robots using neural networks as well as direct mapping.

Natural human robot interaction involving the gestures using the arms of Zeno robot was also developed in this thesis. Zeno is a childlike humanoid robot by Hanson RoboKind. Zeno is used as a robotic platform for research with Autistic children. The robot is taken periodically to UNT Health Science Center for running the experiments with ASD kids and collecting the data. As a part of this research, the scripted motion for the Zeno robot was developed in LabVIEW. This scripted motion of the robot includes a series of gestures including hand wave, tummy rub and fist bump. Each of it performed three times consecutively with both

left and right arm, one arm at a time. The movements of both the Zeno and the robot are captured using a motion capture system and the motions are compared using Dynamic time warping technique to compare how close the child is following the robot.

Figure 1.1 Zeno Robot

As a part of my research, I made some contributions to the DARPA Robotics Challenge (DRC). UT Arlington partnered with RE2, a leading robotics company based in Pittsburgh, PA to compete in the DRC and we were competing in Track B. My contributions to this project were in two areas. 1) Robot tele-operation using the 3D Kinect sensor and 2) Making the ATLAS robot stand up in case it falls down. In the DARPA Robotics Challenge (DRC) competition the ATLAS robot has to perform various tasks including driving a car, connecting a hose and many other task that are generally designed to be operated by humans. Robot tele-operation with Kinect will be a useful tool for many such applications. The person doing the Tele-operation performs the task in front of the Kinect and the algorithm in the computer extracts the joint angles from the Kinect, converts it and maps it to the joint angles of the robot sends it to the controller. The result of this is that the robot mimics the operation of the human in real-time. The other part of

the contribution involves making the robot stand up in the event it falls down. This involves identifying if the robot is lying down in supine or prone position, generating joint position and velocity trajectories that will change the contact states of the robot in a sequential manner, interpolating and filtering these trajectories using cubic spline interpolation and finally feeding these trajectories to the joint controllers of the Atlas robot so that the robot performs the getting up sequence of operations.

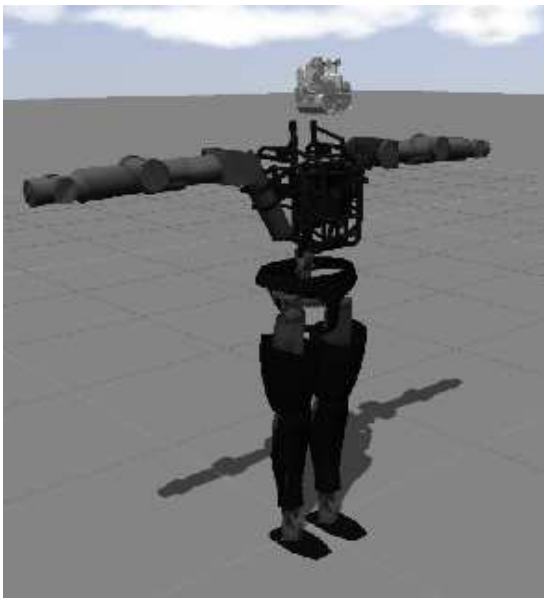


Figure 1.2 Atlas robot

1.4 Research Contributions of Thesis

The primary research contribution of this thesis is in the area of vision-based natural Human robot interaction. This research has been conducted with Zeno robot, PKD android and the Atlas robot in Gazebo simulation.

1.4.1 Natural HRI with Zeno

The research contributions with the Zeno robot includes the development of the gestures like hand wave, tummy rub and fist bump. These gestures were developed for the robot to interact with children diagnosed with Autism. The 3D depth senor Kinect was used to

extract the information from the human joint angles and map it to the robot. With this setup, the root was able to imitate the arm, torso and head motions of the human.

1.4.2 Natural HRI with PKD

The PKD android avatar was used to develop vision-based interaction for face tracking by the android. The android's camera in the eyes was used to detect faces. The research contribution was to develop a novel algorithm to distribute the motions between the neck and the eye servo motors so the neck eye coordination of the android looks more humanistic. Another research contribution was to map the human facial expression to the robot. The PKD android is capable of full range of human facial expressions and the contribution was to map the facial features of the human face to the actuator space of the robots. This mapping was done with both linear way using direct mapping approach and non-linear way using neural networks.

1.4.3 Natural HRI with Atlas Robot

The Atlas robot in the Gazebo simulation was used to develop mapping between the human joints and the Atlas robot. The Atlas is a humanoid robot with 28 degrees of freedom. Here, the research contribution was to develop a novel way of using Quaternions in solving the singularities in the joint angle parameterization that was used for the mapping. This natural interaction with the Atlas robot by imitation of the full range of human motions will enable the robot to be able to tele-operate the robot remotely.

1.5 Thesis Organization

Chapter 2 describes the Literature Review in 1) using robots for diagnosis and treatment of Autism in Kids 2) Social robots and its use 3) Robots imitating the motions of the human and finally 4) Facial expressions mapping to Androids.

Chapter 3 describes the robots used for conducting research in this thesis. It describes the hardware details of the Zeno, PKD and Atlas robot.

Chapter 4 describes the algorithms developed for vision-based natural HRI. It starts with the description of the implementation of Image Based and Position Based Visual Servoing

to control the Pan and Tilt motion of a robot with two degrees of freedom. It then describes the use if Image-Based Visual Servoing technique for the distribution of neck and eye motion of the PKD robot for the movement to appear Humanlike. It also develops algorithms for mapping human motion to that of the robot. In this there are two types, the first one maps the arms and torso motion to the ATLAS robot and the second maps the facial expressions from the human to the PKD android using Linear and Non Linear methods.

Chapter 5 describes Real-time control of Zeno robot for Autism therapy. In this chapter two modes of operation with Zeno is discussed. The first one is the interactive mode where the Zeno robot mimics the motion of the therapist and in the second one it performs a predetermined sequence of gestures that is used for conducting ASD experiments at UNT Health Science Center.

Chapter 6 describes a technique to make the robot stand up from its lying down position. This was done as one of the contributions to DARPA Robotics Challenge.

Chapter 7 finally concludes the thesis with the contributions made and discussed some of the research that could be extended in the future based on the work done in this thesis.

CHAPTER 2

LITERATURE REVIEW

2.1 Research in Autism

Autism Spectral Disorders are characterized by children having difficulty in social interaction with other kids. They tend to shy away from humans and have problems communication in both verbal and nonverbal ways. In the United States, ASD has been seen in 1 of 88 children [6]. There are now increasing evidences that children with Autism may have motor difficulties as well.

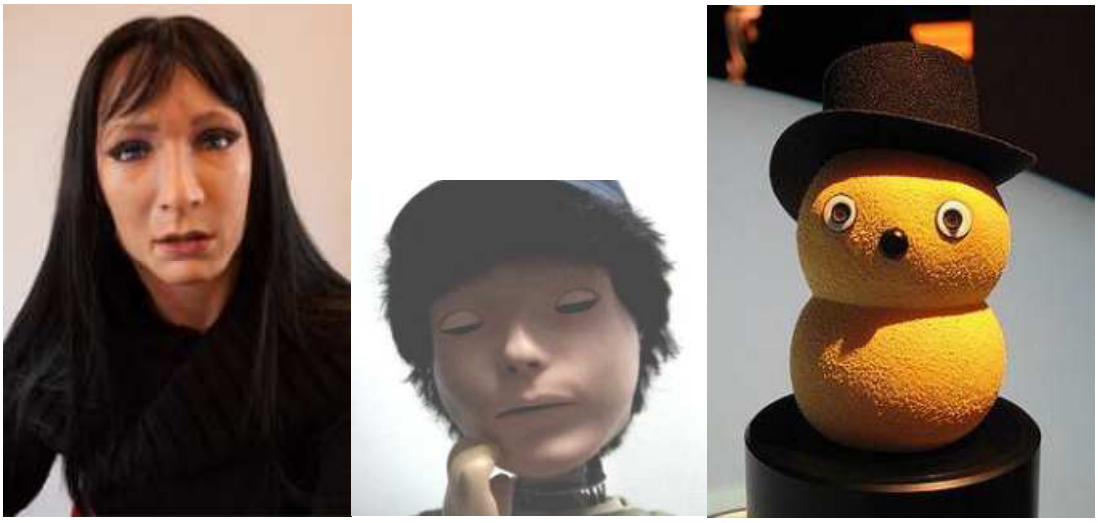
There are some recent works that show a link between the mirror neurons and the Autism [7]. Even though there is still a debate about the link between the mirror neurons and Autism, many studies show a positive link between them. Mirror neurons are helpful in developing the motor skills that are required for social development and interaction. Skills like imitation of facial expression and body gestures are considered an important part of the development of social interaction in children.

Understanding the limitations in the planning and coordination of movement and posture is fundamental to a comprehensive understanding of the impairments and functional limitations link to ASD. Slowed or uncoordinated head and arm movements may limit head turning, reaching, pointing, showing and sharing that are key components of initiation and response to joint attention [8]. A child's poor coordination and slowed movement are linked to poor social participation and increased anxiety during playtime [9][10]. To fully engage in social interaction, a child requires a full repertoire of movement behaviors for use in communication and for understanding the communicative nature of others' movements.

Humans learn new skills using numerous techniques and modalities, in particular through observation and imitation [11][12]. Imitation of motor gestures and movements is

paramount during childhood for the development of social interactions. Contrary to typically developing children, those with ASD do not imitate the movements of other people and do not point towards an object of interest. We believe that their motor imitation difficulties contribute to lack of play and interaction with other children. Children with ASD have difficulties initiating and engaging in imitation behavior.

The advancement and cost-effectiveness of sensor/actuator and computing technology has played an important role in the recent emergence of robotic technology. Recently, use of pattern recognition from sound data as well and the use of human and non-human robots for improving play in children with ASD was considered [13][14][15][16]. Several robotic systems have been developed for use in the therapy of individuals in the autism spectrum such as FACE, AuRoRa, Kaspar, and Keepon, but many of them have only been tested under manual operator control, rather than in a truly autonomous, interactive manner [17]. These studies concluded that the appearance of the robot plays an important role in how children relate and interact with such robots.



(a) (b) (c)

Figure 2.1 (a) FACE (b) KASPER (c) KEEPON [17]

Researchers at Massachusetts Institute of Technology have recently ushered a new research area in social robotics, where the goal is to create realistic interaction between humans and humanoid robots in manners as intuitive and natural as human-to-human conversation [11]. In other words, the humanlike robot hardware has to be controlled so that the overall behaviors are similar to those of humans. People are likely to become familiar with robots that exhibit human-like features, appearance and characteristics. Additional studies confirm that human-like robots that do not have a mechanical-like appearance are treated differently in social settings [18][19]. On contrast, [20] proposed that if the robot appearance are too close to human-like features, an “uncanny valley” is created which leads to feelings of “repulsion” by humans. Additional research is needed to investigate how individuals with ASD respond to robots as instructional agents. However, studies suggest that leading types of interactions useful in engaging children with ASD are emotion recognition, joint attention,imitation and turn-taking [14][15][21].

A robot called Bandit was used to guide older adults to perform imitative exercises [22]. This project had a robot perform upper body gestures that the subject imitated, performance criteria were related to achievement of target poses. These projects and others involving the use of robots for assisting humans in a social, collaborative setting can be considered part of a relatively new field called Socially Assistive Robotics [17]. However, to date, evaluation of human-robot interaction is restricted due to lack of objective criteria that rate imitative gestures and movements and quantify the Human-Robot Interaction (HRI). Moreover, currently available clinical tests that evaluate imitation rely on observation and categorical data of “yes” and “no”. Use of the robot will provide consistent repeatable imitation tasks that can be evaluated using continuous kinematic measures over time. The purpose of this project was to conduct a feasibility study to examine the utility of a robotic device and human robot interaction as a tool for assessing and quantifying imitation behavior in children with autism spectrum disorders.

We aimed to verify if the system and the analysis algorithm can accurately measure imitative behavior in typically developing children and to compare the results with those of children with ASD.

2.2 Social Robotics

The advancements in computing technologies, sensors and actuators have played a major role in the research in social robotics research. There is a lot of research that has been focused on human robot interaction in natural ways similar to how humans interact with each other. In other words, the interface between robots and humans are now controlled to be more humanistic. The motivating factor for our work is to create realistic robots that are capable of interacting in the form of facial expressions, tracking faces, maintaining eye contact to make it more humanistic.

Mori [23] was one of the first to suggest that people are going to be more familiar with robots with human like appearances and if those features are not apparent then it would result in “uncanny valley” which may become repulsive to humans. The study of “uncanny valley” continues to attract research studies. Bartneck [24] proposed the alternative model to the Mori’s “uncanny valley” model, and Shimada [25] suggested that the “uncanny valley” model varies from person to person. If the robots do not have mechanical appearences then the studies done by Woods[26] and Goetz [18] suggested that human-like robots are more likely to be treated the same way people are treated in a social setting. Robots like COG and Kismet from MIT are used extensively to study the human imitation in a social setting with sociable robots. Breazeal [27] discussed the potential challenges regarding building robots that imitate humans, and Scassellati[28] studied human social development models, the way social interaction skills are developed in humans, relating to human-like robot interaction. Gurbuz [29] proposed a novel approach to do human mouth mimicking on a humanoid robot head in real-time. Facial features are tracked in real-time through stereo vision and a learning algorithm is implemented to detect and imitate human mouth movement.

It is clear that creating both a realistic appearance, as well as realistic behaviors for humanoid robots will play an important role in the future of robotics. I chapter 6 we develop a novel algorithm to distribute the motion between the neck and the eyes so that when the robot begins to make eye contact, its eyes and neck move in realistic way as a human moves.

In past work, Breazeal [30] proposed a vision system for social robots that allows them to interact with humans in a meaningful way, e.g., the intention of the human can be perceived and elicit a natural response from the robot. Cannata [31, 32] proposed an approach for tendon driven eyes mechanism based on Listing's law. The assumption for their approach was that the eyes mechanism of the robot is close to that of the human eye. In [33] Rajruangrabin has used reinforcement learning and visual servoing to create head–eye motion profiles that are close to human kinesiology. In [34] a percentage function is used to distribute the motion between the eyes and the neck which may not appear to be realistic movements In chapter 6, a straightforward, real-time based algorithm proposed here that will control the movement of eyes and neck to track faces.

2.3 Human Motion Imitation by Humanoid Robots

Human motion imitation by robots has numerous applications. Robots are able to perform complex problem more efficiently and if the robots has to work in a human environment then it needs to able to imitate human like motion. Human motion imitation is also helpful in the research involving social robots. These robots for example can be used for the research involving diagnosis and treatment of Autism

In recent years, imitation of the motions by the robots has become a hot research topic. Pollard et al [35] have proposed a method to transform a dance captured motion to a motion that the humanoid robot can execute. Nakaoka et al [36] have realized a whole body control of humanoid robot to imitate Jongara-Bushi dance that is a traditional Japanese folk dance. To maintain the dynamical stability of humanoid robot, they control the trajectory of Zero Moment Point (ZMP) [37] to be inside the polygon of support. Safonova et al [38] use also a pre-

recorded human motion to generate optimal motion of the upper body of Sarcos humanoid robot. The function to be minimized is the difference between the recorded and executed motion by the robot. Ruchanurucks et al [39] have proposed a method to optimize upper body motion of humanoid robot in order to imitate a human record motion. Their objective function preserves the main characteristics of the original motion, and at the same time it respects the physical constraints of the humanoid robot. In [40], a neuroscientific inspired approach is presented, which solves imitation learning of cyclic motion with a set of basic motor primitives. These are learned by clustering and dimensionality reduction of visually acquired human motion data. For reproduction, a movement is classified into motor primitives which are played back sequentially. In [41] and [42] methods are introduced, where Hidden Markov Models are trained with a collection of observations of a demonstrated movement. To reproduce a newly observed movement, the observation is recognized based on a set of trained models. With the complying model, a generalization of the recognized movement is generated.

In chapter 5 we develop an algorithm to map human motions to the humanoid robot called Atlas in a simulation environment called Gazebo.

2.4 Research in Facial Expressions Mapping to Androids

Humans are sensitive to facial features when it comes to communication. When we speak, our lips and other facial features move. These speech related motion are subtle, and they are important in face to face communication and humans are very sensitive to them [43][44]. An android needs to display these motions in order to communicate naturally.

Face detection is another important aspect for natural HRI in Androids. The Android PKD must be able to detect faces successfully and deterministically in order to be able to track the face. In this thesis we use “Haar Cascade Classifier” [45] based approach for face detection. In this technique, first, a classifier (namely a cascade of boosted classifiers working with haar-like features) is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative

examples - arbitrary images of the same size. After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

In this thesis we also describe a neural network based algorithm that maps the facial expressions of the human onto the android and does the facial expressions imitation in real-time. To detect the facial expressions we use the faceshift software that uses an extension of Kinect fusion algorithm for robust real time facial expressions tracking[46]. The mapping problem has been addressed using a linear model in the feature space by Ishiguro in [47] but it has several drawbacks such as it needs markers for motion capture; linear model works only when the marker points coincides with the actuator positions; the actuator space may not necessarily be linearly independent; number of markers is limited by the number of actuators. The same problem has been solved assuming a linear relationship between the marker space and the actuator space in [48].

CHAPTER 3

DESCRIPTION OF THE HARDWARE

3.1 Zeno Hardware Description

Zeno is a child-size, 2 foot tall, articulated humanoid robot by Hanson Robotics with an expressive human like face shown in Fig. 31. It has 9 degrees of freedom (DOF) in the upper body and arms, an expressive face with 8 DOF, and a rigid lower body [19]. The robot is capable of moving the upper body using a waist joint, and has four joints in each on the arms implemented using Dynamixel RX-28 servos. It has a 1.6 GHz Intel Atom Z530 processor onboard and is controlled by an external Dell XPS quad core laptop running LabVIEW [20]. The appearance of Zeno is based on a fictitious character - he looks like a 4-7 year old child, and his head is about ¼ the size of an adult human head. Its unique features include life-like skin made of Frubber™ material.

Figure 3.1 Zeno RoboKind R-30

The appearance of Zeno is a game changing experience thanks to this material and to the robot esthetics. The head of Zeno is powered by 9 servo motors, has 3 degrees of freedom (DOF) at the neck joint, and it is capable of panning, tilting the head back and forth as well as left to right. It also has 2 degrees of freedom in each eye (pan and tilt), and 4 of the servos are used for generating facial expressions (eye blink, jaw motion for smile, eyebrow motion for frown, etc.).

Each arm has 4 degrees of freedom, alpha, Beta, gamma and theta as shown in the figure 3.2

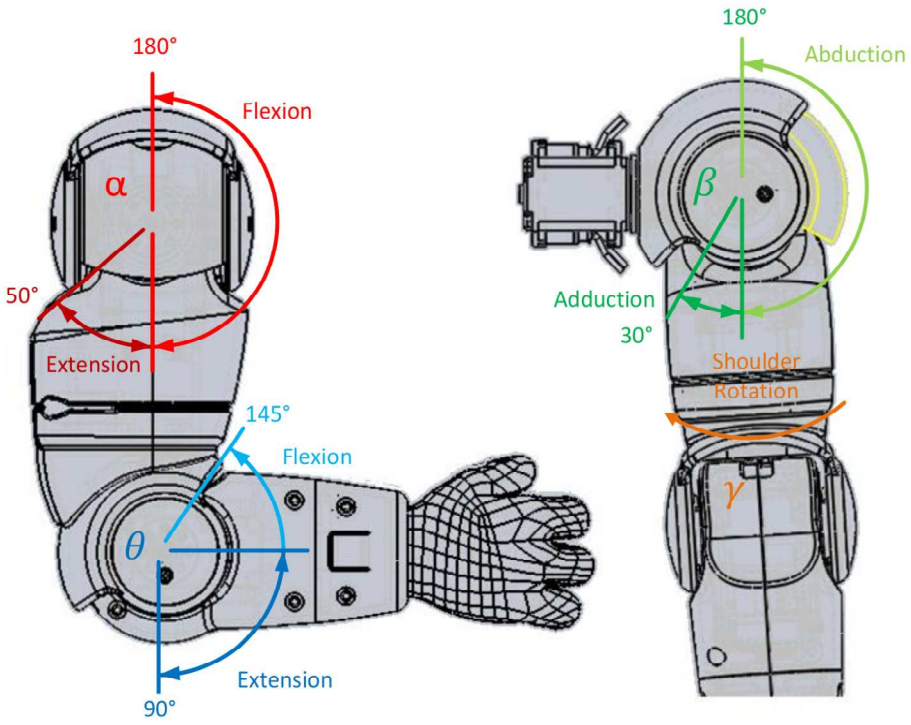
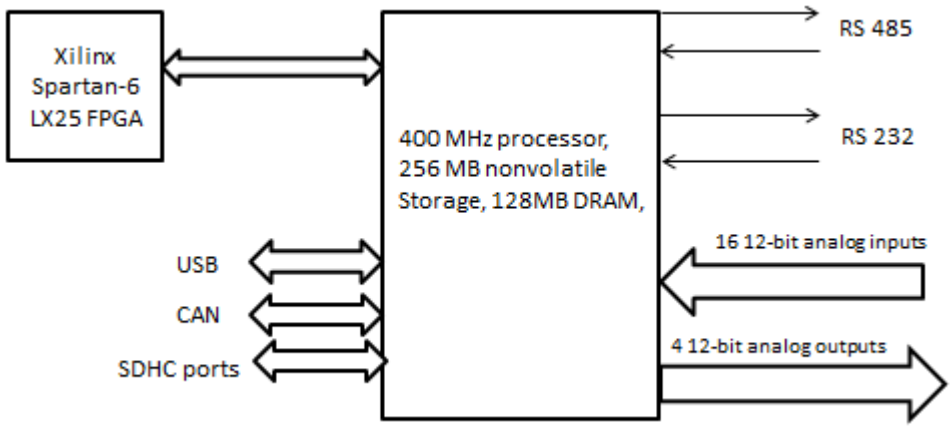


Figure 3.2 Zeno arm configuration [49]

The Dell XPS laptop is used for handling the vision tasks like running the skeleton tracker and performing inverse kinematics on the joint angles to map to the corresponding joints to that of the robot. The control of the robot however is done using the LABVIEW sbRIO 9633 shown in the figure 3.3. This board is suitable to do real-time control as it integrates a real-time processor, a user-reconfigurable FPGA, and I/O on a single printed circuit board (PCB). It

features a 400 MHz industrial processor; a Xilinx Spartan-6 LX25 FPGA; 16 single-ended, 12-bit analog input channels at 500 kS/s; four 12-bit analog output channels; and 28 digital I/O (DIO) lines. The sbRIO-9633 offers a -40 to 85 ℃ local a mbient operating temperature range along with a 9 to 30 VDC power supply input range.

(a)



(b)

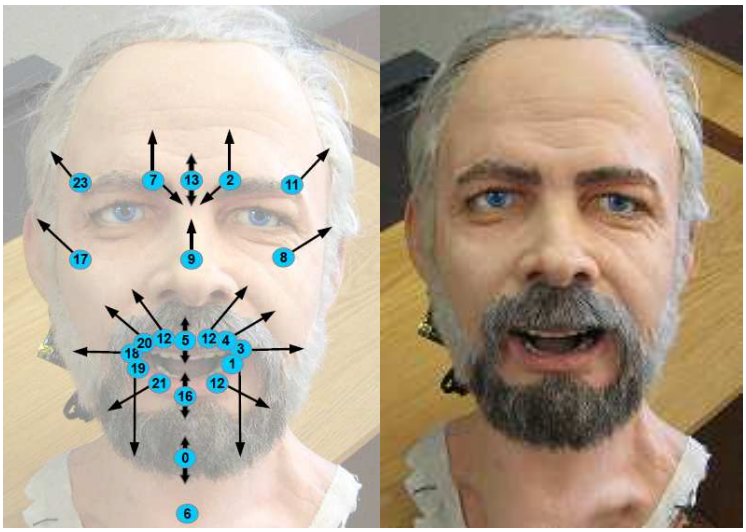
Figure 3.3 (a) sbRIO 9633 board (b) Block diagram of sbRIO 9633 [50]

The communication between the sbRIO and the Dell XPS laptop is established using TCP/IP communications with LABVIEW real-time sharing of variables.

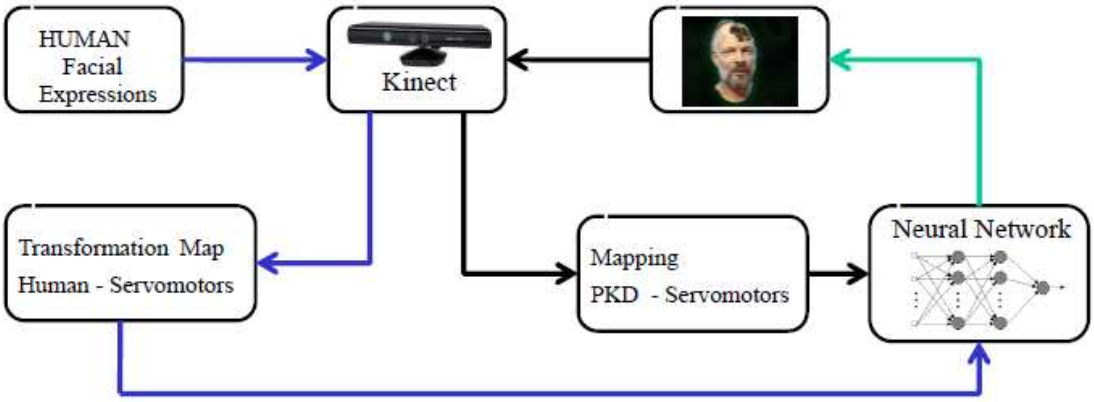
3.2 PKD Android Hardware Description

The android portrait of Philip K. Dick (PKD) is an intelligent robotic resurrection of the late science fiction writer who authored VALIS, Do Androids Dream of Electric Sheep, UBIK, and many other masterworks. In these works, PKD often portrayed robots who thought they were

human and then suffered identity crisis. By portraying PKD in an extremely realistic android, we seek both to explore this theme in a new form of interactive narrative, and also to serve novel research in artificial intelligence and human-robot interaction. The new PKD android, which is used in the described experiments, was built in 2010 with sponsorship from VPRO—Dutch public broadcasting.



(a)



(b)

Figure 3.4 (a) PKD robot with servo motor positions (b) Block diagram of PKD mapping setup

The new PKD android hardware is innovative in several aspects, including novel facial materials, underlying expression mechanisms, and new approaches to aesthetic unification of these into an appealing humanlike robot. First, the robot takes advantage of one of the latest versions of Hanson Robotics' formulations of artificial skin materials known as Frubber. The Frubber material results from bio-inpsired lipid-bilayer interactions that create nano-scale cells in self-assembled networks. These techniques result in a silicone elastomer far more compliant and strong that the base polysiloxane material, which requires 23x less force to deform into expressions. Moreover the resulting expressions fold and crease naturally in ways analogous to human facial soft tissues. The formula for the PKD android explored more radical lipid bilayer techniques and finer exploration of the skin pigmentation for improved sub-surface scattering relative to previous Hanson robots.

Connecting this skin over a skull-like mechanical shell, and to servomotors within the skull form, allows the motors to pull the skin along vectors that correspond to human facial muscle action. In some cases, one motor will replicate the action of two opposing muscles—since muscles only contract, while the motors are reversible. Thus, with 31 degrees of freedom, the PKD android replicates the action of the 48 major muscles in the human face. Given the low-force requirements and high-elongation of the Frubber material, combined with numerous expressions techniques that prevent the multiple expressions DOF from interfering and binding, the PKD android can generate large, human-scale expressions, across the range of possible human facial action units. As far as the authors know, these capabilities of large scale, full-range facial expressions are unique to Hanson Robots.

The degrees of freedom in the PKD android include 4 DOF in the neck (two pitch, one roll, and one yaw), 1 jaw DOF, three eyes DOF (each eye turns independently, but the two are coupled for pitch), upper lids DOF, lower lids DOF, 5 DOF in the forehead simulating the frontalis, eyebrow, and procerus muscle actions, 1 nasalis DOF along the nose, 2 orbicularis oris DOF, to effect Duchenne smile actions, and 12 DOF around the lips. The lips include a

drawstring that will contract the lips, but also allow the lips to expand to the large aperture sizes required for a smile, grimace, or surprise expressions. The numerous other techniques used to generate the expressions are beyond the scope of this paper, but interoperate to achieve the lifelike actions of the PKD android face.

To control the facial expressions, a local notebook computer sends commands to the servo motors via USB signals sent to a Pololu Maestro servo controller, which generates the PWM motion commands for the individual servo motors. The Pololu servo controller used is shown in the picture below

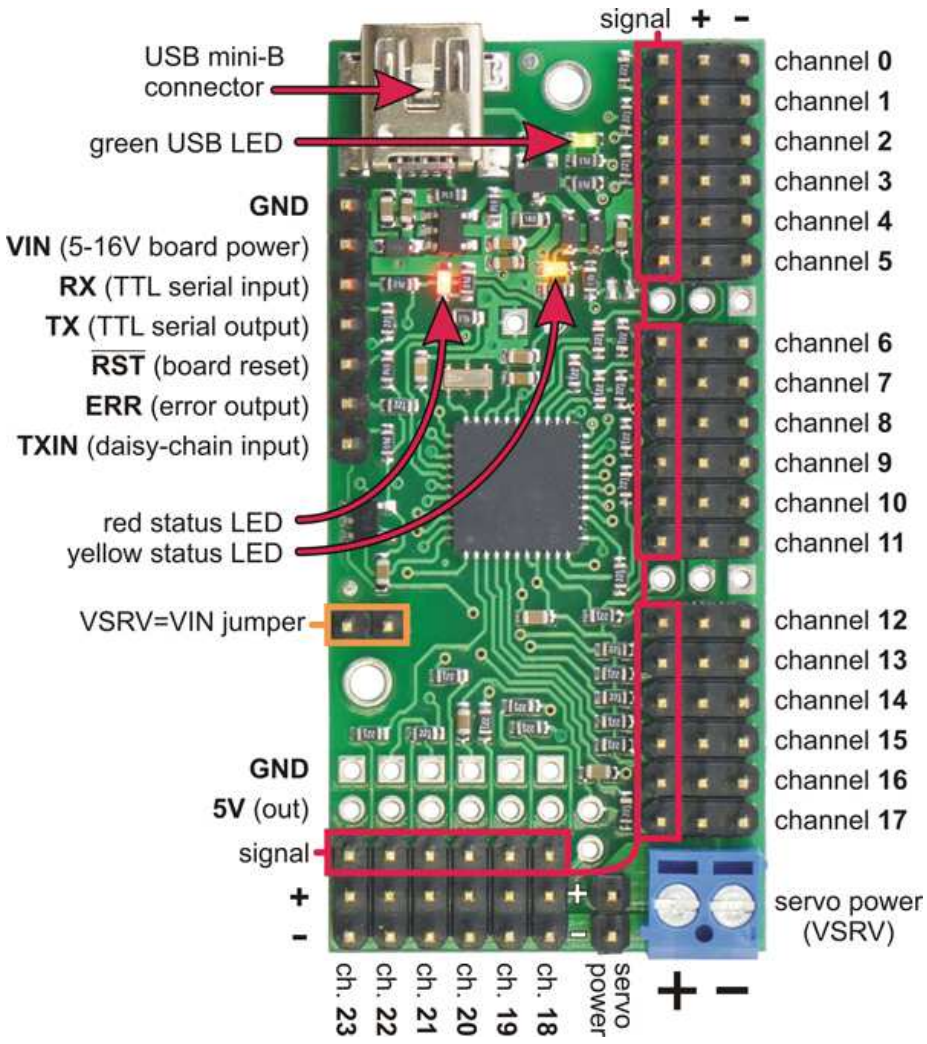


Figure 3.5 24 channel Pololu servo controller [51]

3.3 Description of the Atlas Robot

ATLAS is a hydraulically powered robot in the form of an adult human. It is capable of a variety of natural movements, including dynamic walking, calisthenics and user-programmed behavior. Based on the Petman humanoid robot platform, Atlas was modified to meet the needs of the DARPA Robotics Challenge [52].

- Near-human anthropometry
- 2 arms, 2 legs, torso and head.
- 28 hydraulically actuated joints with closed-loop position & force control
- On-board real-time control computer
- Electric power & network tether
- On-board hydraulic pump & thermal management
- Crash protection
- Modular wrists accept 3rd party hands
- Head-mounted sensor package with LIDAR, stereo sensors, dedicated sensor electronics and perception algorithms.

The Atlas robot described above is simulated in a Gazebo simulation environment. Fig 1.2 shows the Atlas robot in the Gazebo simulation environment. We use this simulated robot to our research purpose. Gazebo is a multi-robot simulator for outdoor and indoor environments. Like Stage (part of the Player project), Gazebo is capable of simulating a population of robots, sensors and objects, but does so in a three-dimensional world. It generates both realistic sensor feedback and physically-plausible interactions between objects (it includes an accurate simulation of rigid-body physics) [53]. Some of the outstanding features of Gazebo are:

- Access to multiple physics engines including ODE and Bullet
- Access to parameters such as accuracy, performance and dynamic properties of each model
- Using OGRE, Gazebo provides realistic rendering of environments

- State of the art GPU shaders generate correct lighting and shadows for improved realism
- Generate sensor information from laser range finders, 2D cameras, Kinect-style sensors, and RFID sensors
- Many robot models are provided including PR2, Pioneer2 DX, iRobot Create, TurtleBot, and generic arms and grippers
- Support for ROS

Gazebo is the basis for the DRC Simulator, which is used by participants in the DRC program, especially the Virtual Robotics Challenge (VRC), a cloud-based simulation competition scheduled for June 2013 [54]. Building upon Gazebo, the DRC Simulator, or DRCSim, comprises additional worlds, models, and code that is specifically required to simulate the DRC domain. Robot Operating System (ROS) fuerte in Ubuntu Linux 12.04 is used as a Software platform for programming.

CHAPTER 4

VISION-BASED NATURAL HUMAN ROBOT INTERACTION

4.1 Introduction

In this chapter we consider the development of algorithms for vision-based natural human robot interaction. Vision is normally used as a feedback mechanism to achieve certain tasks. In this chapter we use the vision feedback in many areas of human robot interaction. Mimicking of human motions by the robots is one of the major aspects that are focused in this chapter and it also plays an important role in human robot interaction. By making a robot imitate the body movements and facial features of the human we can make the robot more sociable and being accepted by the humans. This mimicking could be the motion of the hands and the torso or it could be mimicking of the facial expressions of a human. The input device that is often used for sensing the current state is the Microsoft Kinect sensor. The low cost of this device makes the depth sensing accessible to many users. We use the Kinect sensor to track the skeletal coordinates of the human. This skeleton tracking algorithm is used form the NITE middleware from OpenNI [55]. The research area that is addressed in this chapter is how can we map the dataset processed using the Kinect and nite algorithm to the actuator space of the robot in order for the mimicking to appear more natural. The facial expressions are another area that is addressed in this chapter for mimicking. We use the Kinect sensor and faceshift software to extract the facial features of the human. These facial features are then mapped to the servo actuator space using direct mapping with scaling and using neural networks.

Cameras are another input device that is used a feedback device in vision-based control systems. In this chapter we use camera as a feedback mechanism to find the position of the face in the image. The control algorithm that is then developed to servo the camera to

position the face in the center of the image. This type of control is called visual servoing. This kind of human robot interaction is useful for the robot to make eye contact with the human that it is interacting. Visual servoing is the use of computer vision to control the motion of the robot. Here there are two basic type of approaches namely position based visual servoing (PBVS) and image based visual servoing. In the image based visual servoing, we use the features of the image to generate the error signal and in PBVS we use the vision data to extract the features in the 3D space and generate the error signal. For controlling the neck and eye motion of the android PKD we first consider the implementation of pan and tilt motion with two servos and applying both the PBVS [56] and IBVS [57] techniques and compare the results. Based on the results of this experiment we then develop an algorithm to distribute the motion between the neck and the eyes for tracking faces by the android PKD with realistic humanlike movements.

4.2 Image based Visual Servoing with Pan and Tilt Motors

In this project we use a robot with two degrees of freedom namely Pan and Tilt and we mount a camera on it for tracking a person and 3D position estimation. This robot is shown in the figure 4.1 below.

Figure 4.1 Robot with two degrees of freedom (Pan and tilt)

For face detection we use the OpenCV ‘Haar Feature-based Cascade classifier ‘algorithm [45]. This algorithm works in two stages: training and detection. In the training data preparation phase we need a set of samples. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to detected objects. Set of negative samples must be prepared manually, whereas set of positive samples is created using ‘opencv_createsamples’ utility. After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a "1" if the region is likely to show the object (i.e., face), and "0" otherwise. To search for the object in the whole image we can move the search window across the image and check every location using the classifier. This classifier is designed in such a way that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

Once the face is detected, next step is to compute the error to implement the PID control for tracking the face. The error is calculated as the difference between the center of the image to the center of the face. This error is given as

$$e(t) = I_c - I_c^*. \tag{4.1}$$

where,

Ic is the center of the image and I_c^* is the center of the face detected in the image.

4.2.1 Camera Calibration

In order to figure out the center of the image, we used camera calibration algorithm from OpenCV. Using this algorithm we also succeed to determine the relation between the camera’s natural units (pixels) and the real world units (meters). The camera has some significant distortion, but these are constant and with a calibration and some remapping we

managed to correct it. For the distortion OpenCV takes into account the radial and tangential factors. For the radial we have used the following formula [58]:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

(4.2)

So for an old pixel point at (x, y) coordinate in the input image, for a corrected output image its position will be (X-corrected Y-corrected). Tangential distortion occurs because the image taking lenses are not perfectly parallel to the imaging plane. Correcting this is made via the formulas:

$$x_{corrected} = x + (2p_1 y + p_2(r^2 + 2x^2)$$
$$y_{corrected} = y + (2p_2 x + 2p_1(r^2 + 2y^2)$$

(4.3)

So we have five distortion parameters, which in OpenCV are organized in a 5-column one-row matrix:

$$Distortion_{coefficients} = [k_1\ k_2\ p_1\ p_2\ k_3]$$

(4.4)

Now for the unit conversion, we have used the following formula:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & I_x \\ 0 & f_y & I_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

(4.5)

where f_x and f_y are the focal lengths, $(I_x\ I_y)$ what are the optical centers expressed in pixels coordinates and $[k_1\ k_2\ p_1\ p_2\ k_3]$ are the Distortion coefficients

If for both axes a common focal length is used with a given aspect ratio (usually 1), then $f_y = f_x * a$ and in the upper formula we will have a single focal length. The matrix containing these four parameters is referred to as the camera calibration matrix. While the distortion coefficients are the same regardless of the camera resolutions used, these should be scaled

along with the current resolution from the calibrated resolution. The process of determining these two matrices is the calibration. Calculating these parameters is done by some basic geometrical equations. The equations used depend on the calibrating objects.

For the calibration purpose we used the Classical black-white chessboard shown in the figure 4.2. For more details on conducting the experiment refer the documentation from OpenCV[58]. Basically we took some snapshots of this chessboard with setup camera and ran it through OpenCV. Each found pattern equals in a new equation. To solve the equation we used a predetermined number of pattern snapshots to form a well-posed equation system. The chessboard we used requires at least two equations. However, we ended up having a good amount of noise present in our input images, so for good results we took 30 good snapshots of the input pattern in different position. Since the calibration needs to be only once per camera, we saved the parameters after we got the successful calibration. The parameters output by the camera calibration are the distortion coefficients, focal length and the optical center.

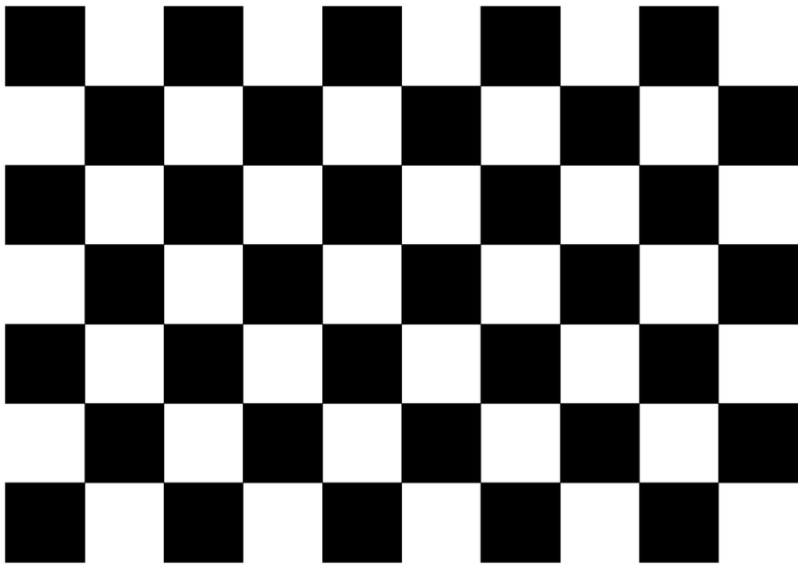


Figure 4.2 Chessboard pattern used for camera calibration

4.2.2 Control Algorithm

The error $e(t)$ computed as the difference between the optical center and the center of the persons face detected in OpenCV. After we have the error $e(t)$, we implement a PID control to make the error converge to zero. The input $u(t)$ then is given by :

$$u(t) = K_p e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t).$$

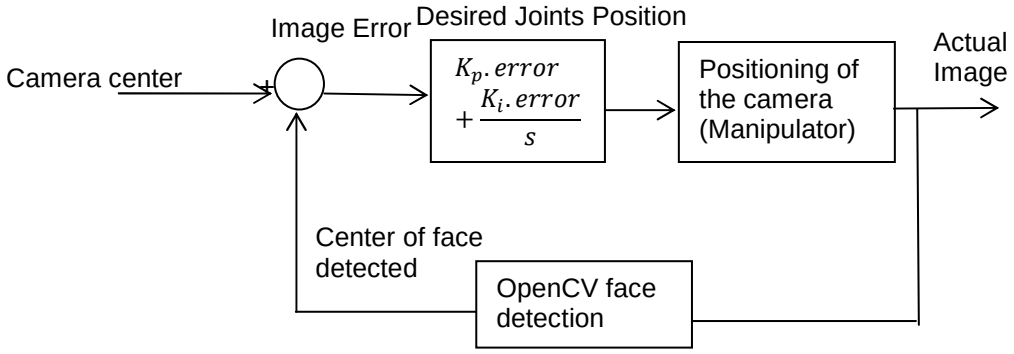


Figure 4.3 Block diagram for the IBVS method

The K_p, K_I, K_D gains are tuned using Zeigler Nichols method to minimize the rise time and overshoot so that the robot is responsive. Initially K_I and K_D are set to zero and K_p is increased until the camera started oscillating with constant amplitude and this gain is called K_u. The oscillation period T_u is used for P I and D gains using the formula

$$K_p = 0.45K_u$$

$$K_i = \frac{2K_p}{T_u}$$

$$K_D = \frac{T_u}{8K_p} \tag{4.6}$$

The values found for K_p, K_i, K_D through this method are 0.2, 1, and 0.1.

4.3 Position based Visual Servoing using Pan and Tilt Motion

PBVS can also be referred as Kalman filter based 3D position estimation and tracking. As sensors (camera in our case) are uncertain and models are always incomplete, estimation theory is at the core of many problems in robotics. And in such cases tools like Kalman filter gives a powerful framework in which we can progress. Position-based control schemes use the pose of the camera with respect to some reference coordinate frame to define s. Computing that pose from a set of measurements in one image necessitates the camera intrinsic parameters and the 3-D model of the object observed to be known. This classical computer vision problem is called the 3-D localization problem. Basically in PBVS method, the 3-D position of an object (human face) with respect to camera reference frame is used to control the servomotors.

For the face detection purpose, we have used the same OpenCV 'Haar Feature-based Cascade Classifier' algorithm as explained in the section 4.1.1. The object coordinates are related to the image frame through a standard transformation written as:

$$v = fk_v \frac{y_c}{z_c} + v_0 \tag{4.7}$$

v is the radius of the face detected in the image plane in pixels, focal length f, k_v, v_0 are the intrinsic camera calibration parameters. Y_c is the approximate radius of the average human face, usually considered as 3.03 inch. Z_c is the only unknown factor in the above equation and is the depth of the object i.e. the distance between the object and the camera. Once we have the $Z_c$using the above equation, we can find out X_c, horizontal distance travelled by the object using the following formula:

$$u = fk_u \frac{X_c}{Z_c} + u_0, \text{ or}$$

$$X_c = \frac{(u - u_0)Z_c}{fk_u}.$$

(4.8)

Here u is in pixels and it is the center of detected human face in the camera. fKu and $u0$ are the intrinsic camera calibration parameters estimated as described before; Zc is calculated as shown before.

Now we measure the velocity in X and Y directions the using the formula:

$$\dot{X} = \frac{X_t - X_{t-1}}{T}$$

$$\dot{Y} = \frac{Y - Y_{t-1}}{T}$$

$$\dot{Z} = 0$$

(4.9)

The measurement dZ/dt is 0 since there is no dynamics considered here for the motion in the Z direction.

The Kalman Filter requires a dynamical model and a measurement model. For the dynamical model, it is assumed that the object moves with a constant velocity over a defined sampling time, which is certainly the case at low speeds, typical of natural human motions during conversations with a robot. The measurement model for the Kalman Filter is given by (7) to (9), as detailed before.

4.3.1 Position Vector Estimation

For the estimation of position vector, the dynamic model is defined as constant instantaneous velocity as follows:

$$\ddot{S} = 0$$

(4.10)

$S = [X\ Y\ Z\ \dot{X}\ \dot{Y}\dot{Z}]$ is the state variable. So the discrete model of the Kalman filter is given as

$$S_{k+1} = AS_k + \gamma$$

(4.11)

Where A is given by

$$A = \begin{bmatrix} O_3 & TxI_3 \\ O_3 & I_3 \end{bmatrix}$$

(4.12)

I3 and O3 are 3x3 identity and zero matrices respectively, T3 is a 3x3 matrix which has the time difference between two successive predictions and γ is the process noise of the discrete time dynamical model described as Gaussian with zero mean and a noise covariance of Q.

The Kalman filter used is described by the following model.

Model:

$$S_{k+1} = AS_k + \gamma$$

$$B_k = HS_k + \vartheta_k$$

(4.13)

Where $B_k = [\, X\ Y\ Z\ \dot{X}\ \dot{Y}\ \dot{Z}]$ which is measured as described using the equations (1) through (3) and H is a 12 x 12 identity matrix.

Gain:

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$$

(4.14)

where

$$H_k = \frac{\partial h_k}{\partial S_k}|_{S_k^-}$$

(4.15)

Prediction:

$$\hat{S}_{k+1}^- = D\hat{S}_k^-$$

$$P_{k+1}^- = AP_k^+ A^T + Q_k$$

(4.16)

Update:

$$P_k^+ = [I - K_k H_k] P_k^-,$$

$$\hat{S}_k^+ = \hat{S}_k^- + K_k\left[b_k - h_k\left(\hat{S}_k^-\right)\right].$$

(4.17)

Once the position is estimated we calculate the equivalent PWM signal by using a linear mapping and send it as commands to the servo controllers so that the Pan and tilt motors will align itself towards the persons face.

4.3.2 Results

To check the accuracy of the 3D position estimation we collected the estimated data and the actual value (Ground truth) and compared it with each other and found that the estimation is accurate when the person is near to the camera and as the person tends to go far away the difference between the estimated value and the ground truth starts to grow. This is clear as we do not have any prior in the Z direction as we are not modeling the dynamics in Z direction. Therefore we rely on the measurement of the radius of the face alone to estimate the depth which is very sensitive as a small radius change results in large changes in the distance. The figure 4.3 illustrates this. The X, Y and Z axis represents the distance from the camera. For each point we have the ground truth and the estimated value.

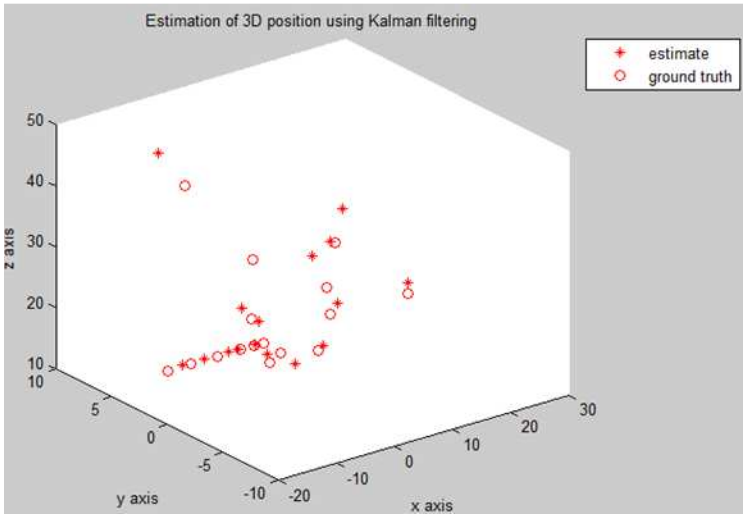


Figure 4.4 Estimation of the position in 3D vs. the ground truth

4.4 Distribution of Motion to the Eyes and Neck Servo Motors for the Movement to Appear Humanlike

The PKD robot has a CMOS camera in its right eye and we use it to track faces. The eyes have 3 degrees of freedom together. Individual eyes can pan independently but the tilt (up/down motion) is actuated by a common actuator. The neck also has 3 degrees of freedom that can pan, tilt and rotate. We use image based visual servoing for the robot to track faces. For tracking purposes we can consider the error to be the difference between the center of the image and the center of the face/object detected in the image.

$$e_x = x^* - x^f$$

$$e_y = y^* - y^f$$

(4.18)

For each of the above error we can either move the neck or the eyes to make the error zero. However, in order to appear the motion to be more realistic like the way humans move their eyes and neck we need to control the speed of eyes and neck movement according to the distance between the robot and the object it is tracking. If the robot is tracking an object that is close to the robot then the eyes should move quickly to track the object and the neck will follow slowly. If the object that is tracked is far then neck movement becomes predominant

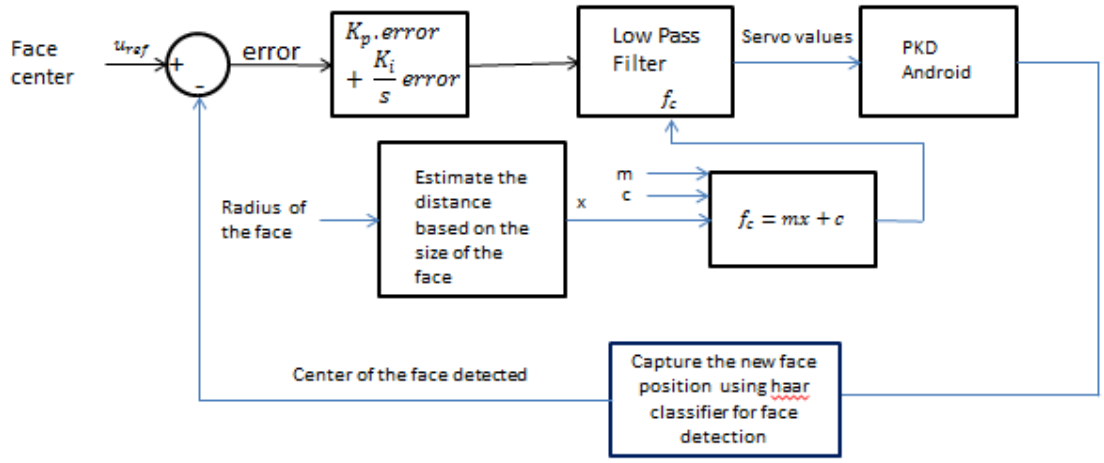


Figure 4.5 Block diagram for the motion distribution between neck and eye servos

We use PI control for controlling the eyes and neck actuators. The eyes are controlled to make the errors ex and ey zero

$$P_x = K_{Px}e_x + K_{Ix}\int_0^t e_x dx$$

$$P_y = K_{Py}e_y + K_{Iy}\int_0^t e_y dy$$

(4.18)

where,

K_{Px} , K_{Py} are the proportional gains of the x axis controller,

K_{Ix} , K_{Iy} are the integral gains of the y axis controller,

P_x , P_y are the servo values for the Pan and tilt motors ranging from 0-255

The eye controllers have high Kp and Ki gains so that they are able to move quickly and are also carefully calibrated with trial and error so that the overshoot is minimized.

Similar PI control equations can be generated for the neck motion pan and tilt. For the neck motion the error signal is now the difference between the current servo position of the eyes and the center of the eyes. The error values for x and y are calculated with respect to the x and y center respectively.

$$e_{Nx} = P_x - P_{Cx}$$

$$\mathrm{e_{Ny}} = \mathrm{P_y} - \mathrm{P_{Cy}}.$$

(4.19)

where,

P_x , P_y are the current position of the servo motors in the x and y direction,

P_{Cx} , P_{Cy} are the center positions of the servo motors for the eyes in x and y direction respectively.

A similar PI controller is implemented to minimize these errors which mean that the objective of the eye controller is to move the neck so that the eyes get positioned to the center. The gains of these Kp and Ki values of the servo motors are tuned to low values so that it is not as fast as the eye movements. The control signals to the eyes and neck are then passed to low pass filters. The cut off frequency is varied according to the distance between the robot and the object that is tracked. In case of face tracking this can be measured by the dimensions of the face. If the object is near then the low pass cut off frequency for the eyes are tuned to high values so that the eye movements will be fast and the cut off frequency of the neck will be low values so that neck movements will be slow. For tracking distant objects we vary the cut off frequency linearly based on the distance such that neck motion becomes predominant to track the object. In this way we can track the human face in most realistic way.

4.4.1 Results

The error value defined as the difference between the center of the face and the center of the camera pixels is recorded over time and we can see that the controller does an effective job by making the error converge to zero when there is a deviation.

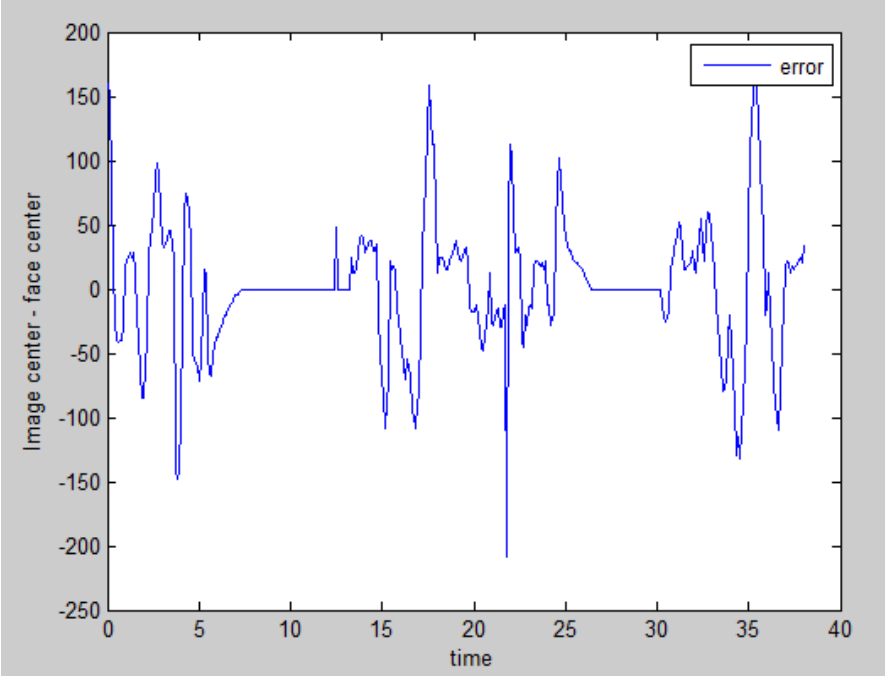


Figure 4.6 Plot of error values e_x of the eye pan servo motors vs. time

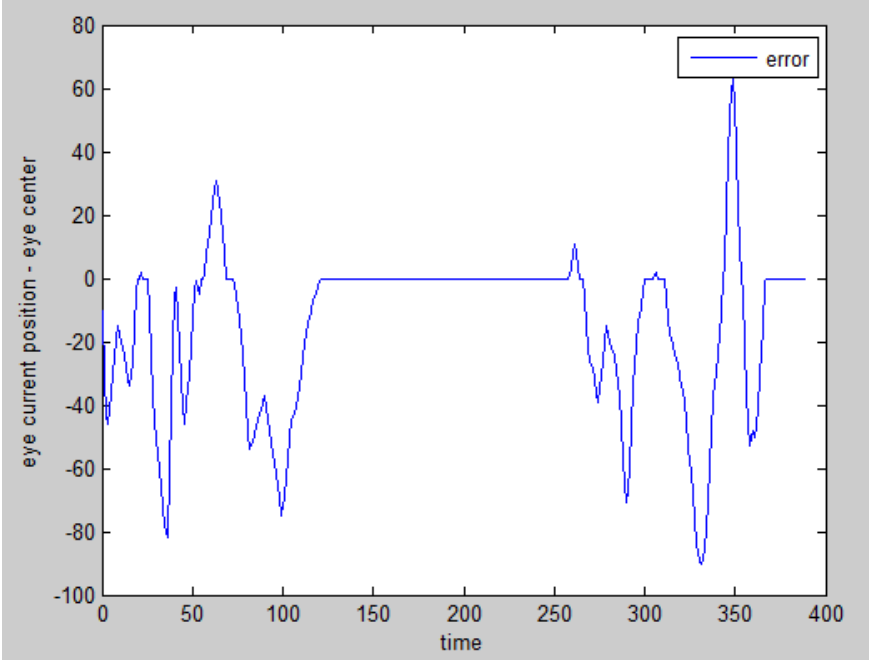


Figure 4.7 Plot of error values $e_y$of the eye tilt servo motors vs. time

4.5 Mimicking the Arm Movements of the Human by the Atlas Robot using Kinect

This project was done as a part of the contribution for the DARPA Robotics Challenge. The primary technical goal of the DRC is to develop ground robots capable of executing complex tasks in dangerous, degraded, human-engineered environments. Competitors in the DRC are expected to focus on robots that can use standard tools and equipment commonly available in human environments, ranging from hand tools to vehicles, with an emphasis on adaptability to tools with diverse specifications. The robot that is used in the competition is called the Atlas robot. Fig 1.2 shows the Atlas robot in the Gazebo simulation environment. Atlas is being developed by Boston Dynamics, Inc. Based on its Petman humanoid robot platform; Atlas is modified to meet the needs of the DARPA Robotics Challenge.

My contribution to this project was to perform tele-operation of the Atlas robot using the 3D Kinect sensor. In the competition the robot has to perform various tasks including driving a car, connecting a hose and many other tasks that are generally designed to be operated by humans. Robot tele-operation with Kinect will be a useful tool for many such applications.

Since the robot needs to operate on various tools designed for human use, I developed a mechanism through which the robot can be tele-operated through Kinect. The software utilizes the skeleton tracking algorithm from OpenNI. This algorithm attaches a transfer frame at each joints of the human and the rotations are modeled using rigid body motion. We can then use parameterization of angles like Euler angles, Roll pitch yaw or Quaternions to extract the joint angles. These joint angles are then filtered using cubic spline filtering to remove high frequency noise and to generate velocity trajectories. These joint angles are then mapped to the Atlas robot joint angles that are then used as set points for the joint controllers so eventually the robot mimics what a human does in real-time.

The Kinect camera gives us the 3D depth information of the surrounding environment. OpenNI provides various application programming interfaces using the raw sensor data from

the Kinect. One such algorithm is the skeleton tracker of a human. The algorithm attaches the Cartesian frames to each of the joints. Fig 4.8 shows the transfer frames attached to the human skeleton.

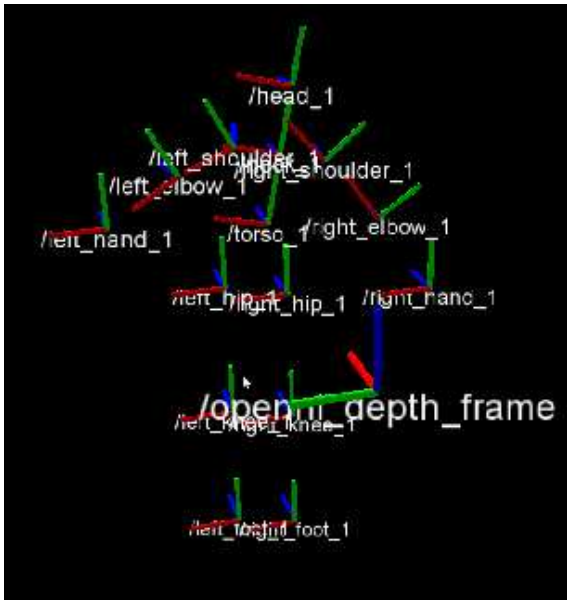


Figure 4.8 Rigid body transfer frames at each joint of the human skeleton.

The TF library in ROS provides an efficient ways to perform rigid body motion transformations. It provides libraries through which we can use various parameterizations like Euler XYZ, Euler ZYZ, roll pitch yaw, Quaternion representations and many more. To extract the human motion joint angles, we need to choose the parameterization such that there are no singularities in the workspace of the joint angles. For the shoulder joint angles for instance we can use Euler roll pitch yaw parameterizations because the range of motion of the human shoulder in approximately limited to $(-\pi/2\ ,\pi/2)$. In the Roll-Pitch- Yaw representation, a series of three ordered right-handed rotations is needed to coalesce the reference frame with the rigid-body frame. Specifically, the reference frame is rotated:

i) about the $ZR = [0\ 0\ 1]'$ axis by an angle $\emptyset$ so that the plane (XR, ZR) contains XE; then

ii) about the new $YR = [-sin\emptyset\ cos\emptyset\ 0]$ axis by an angle θ so that $XR = XE$; and finally

iii) about the new $XR = [\ cos\emptyset cos\theta\ sin\emptyset cos\theta\ -sin\theta\ ]'$ axis by an angle ψ so that $ZR = ZE$

$$R_0^1 = R_z(\phi)R_y(\theta)R_x(\psi)$$

$$R_0^1 = \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}$$

$$R_0^1 = \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}$$

(4.20)

The inverse transformation can be readily computed from the rotation matrix by observing that

$$r_{11} = cos\phi cos\theta$$

$$r_{21} = sin\phi cos\theta$$

$$r_{31} = -sin\theta$$

$$r_{32} = cos\theta sin\psi$$

$$r_{33} = cos\theta cos\psi$$

(4.21)

This inverse transformation problem has two solutions when $cos\theta \neq 0$. The solution for θ in the range $(-\pi/2\ , \pi/2)$ is

$$\phi = Atan2(r_{21}, r_{11})$$

$$\theta = Atan2(-r_{31}, -\sqrt{r_{32}^2 + r_{33}^2})$$

$$\psi = Atan2(r_{32}, r_{33}).$$

(4.22)

The other equivalent solution for θ in the range $(\pi/2\ , 3\pi/2)$ is

$$\phi = Atan2(-r_{21}, -r_{33})$$

$$\theta = Atan2(-r_{31}, -\sqrt{r_{32}^2 + r_{33}^2})$$

$$\psi = Atan2(-r_{32}, -r_{33}).$$

(4.23)

Both the above solutions degenerate when $c\theta = 0$.

For representing roll pitch yaw of the shoulder joint this is not a problem as the human joint angles for the roll pitch yaw tend to be in the open interval (-90 90). Therefore joints involving roll pitch and yaw motion like the shoulder joints, neck joints, torso and hip joints we can use the Euler Roll Pitch and Yaw representations without getting into singularities.

However, when we have to extract the joint angles of the elbow and the knee joint we cannot use the Euler Roll Pitch Yaw parameterization since the range of the elbow and knee joint angles is [0 180). Additionally, the elbow and knee joint has only one axis of rotation. The solution to this is to use the Quaternion representation of the joint angles. Any rotation in three dimensions can be represented as a combination of an axis vector and an angle of rotation. Quaternions give a simple way to encode this axis-angle representation in four numbers and

apply the corresponding rotation to a position vector representing a point relative to the origin in R3. A Quaternion $Q = \{\eta, \epsilon\}$ where

$$\eta = cos\frac{\theta}{2}$$

$$\epsilon = sin\frac{\theta}{2}r;$$

(4.24)

η is called the scalar part of the quaternion while $\epsilon = [\epsilon_x\ \epsilon_y\ \epsilon_z]^T$ is called the vector part of the quaternion. They are constrained by the condition

$$\eta^2 + \epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2 = 1,$$

(4.25)

hence, the name unit quaternion. Unlike the angle/axis representation, a rotation by $-\theta$ about −r gives the same quaternion as that associated with a rotation by θ about r; this solves the above nonuniqueness problem. The rotation matrix corresponding to a given quaternion takes on the form

$$R(\eta, \epsilon) = \begin{bmatrix} 2(\eta^2 + \epsilon_x^2) - 1 & 2(\epsilon_x\epsilon_y - \eta\epsilon_z) & 2(\epsilon_x\epsilon_z + \eta\epsilon_y) \\ 2(\epsilon_x\epsilon_y + \eta\epsilon_z) & 2\left(\eta^2 + \epsilon_y^2\right) - 1 & 2(\epsilon_y\epsilon_z - \eta\epsilon_x) \\ 2(\epsilon_x\epsilon_z - \eta\epsilon_y) & 2(\epsilon_y\epsilon_z + \eta\epsilon_x) & 2(\eta^2 + \epsilon_z^2) - 1 \end{bmatrix}.$$

(4.26)

The inverse problem for a given rotation matrix R, where

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

(4.27)

The inverse problem can now be solved by

$$\eta = \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1}$$

$$\epsilon = \frac{1}{2}\begin{bmatrix} sgn(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ sgn(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ sgn(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix}.$$

(4.28)

We can safely ignore the axis of rotation as it does not change for the elbow joint relative to the shoulder joint. The transfer function library in ROS allows us to use the Quaternion representation and to extract the joint angles.

After successfully extracting the joint angles of the human motion through Kinect hardware and using the algorithm just described, the task now is to map these joint angles to the Atlas robot joint angles. The mappings for most joints are straight forward. We need to scale the joint angles and apply the appropriate joint limits as described in the URDF file of the Atlas robot. The URDF file is where we can find the joint limits and the velocity limits of each motor of the Atlas robot.

The angles after mapping are published to the joint controllers of the Atlas robot which then uses the high gain PID control to move the joints to the desired position and the velocities. Please refer to Appendix C for the Software implementation and code snippets for the algorithm.

4.6 Mimicking the Facial Expressions, Eye Motion and the Neck Rotations of a Human by Android PKD in Real-Time using Kinect

The fig 4.9 shows the positions of the servo motors in the PKD head. It has a total of 31 servo motors.

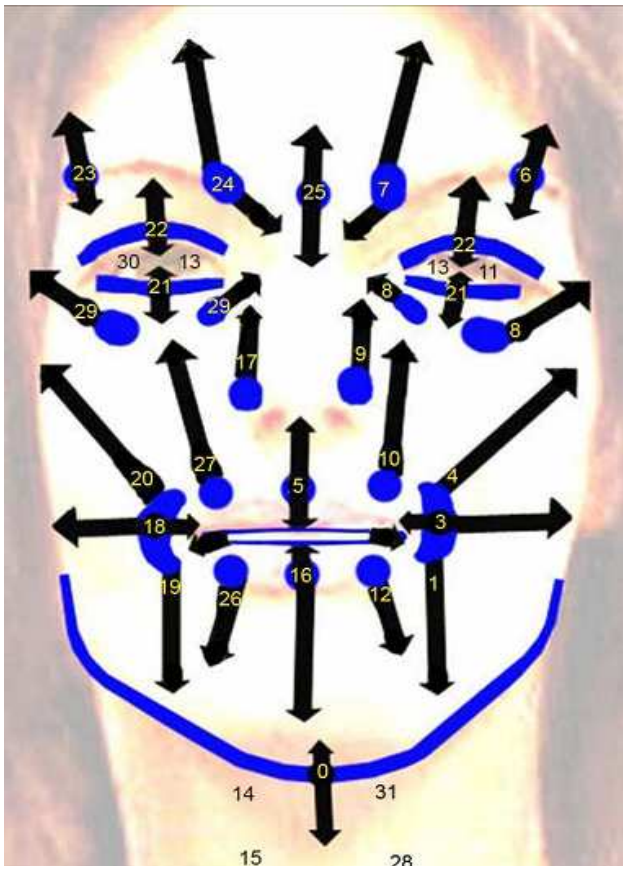


Figure 4.9 Servo motor positions on the android PKD

The goal here is to map the facial expressions of a human to the corresponding servo positions of the PKD so that the expression can be easily copied to the robot. In order to do this mapping first we need to be able to track the human facial expressions accurately. Humans have control on some specific areas of muscles on the face using which he or she will generate the facial expression. We need to be able to track exactly the same set of positions in-order to be able track the facial expressions. Faceshift is a commercial tool that can be used to track the expression of human faces. Faceshift can track with high accuracy because it learns the individuals personalized avatar. The avatar is created from a few training expressions and the resultant tracking is very stable, accurate and expressive. It provides 48 blendshape parameters, which allows us to capture even small emotions. Faceshift also transmits these 48 blendshape variables through a TCP/IP connection. For our purposes we establish a TCP/IP

socket and receive the 48 blendshape variables in real-time for mapping to the servo actuators of the PKD.

We solve the mapping of the Human facial features captured using Faceshift and the PKD servo motors using two approaches. The first one is a direct mapping and second one is an automated using the neural network approach.

4.6.1 Direct Mapping

In the direct mapping, we receive the 48 blendshape variables as shown in the table below. All 48 variables are normalized to the range between 0 and 1.

Table 4.1 Blend shapes variable description

Blendshape names	**Description**	**Equivalent motor position on PKD**
EyeBlink_L	Eye blink left	21,22
EyeBlink_R	Eye blink right	21,22
EyeSquint_L	Eye squint left	NA
EyeSquint_R	Eye squint right	NA
EyeDown_L	Eye down left	13
EyeDown_R	Eye down right	13
EyeIn_L	Eye line left	30
EyeIn_R	Eye line right	11
EyeOpen_L	Eye open left	22,21
EyeOpen_R	Eye open right	22,21
EyeOut_L	Eye out left	29
EyeOut_R	Eye out right	8
EyeUp_L	Eye up left	13
EyeUp_R	Eye up right	13
BrowsD_L	Brows down left	23,24

Table 4.1 - *Continued*

BrowsD_R	Brows down right	7,6
BrowsU_C	Brows up center	25
BrowsU_L	Brows up left	23,24
BrowsU_R	Brows up right	7,6
JawFwd	Jaw forward	NA
JawLeft	Jaw left	NA
JawOpen	Jaw open	0
JawRight	Jaw right	NA
MouthLeft	Mouth left	18
MouthRight	Mouth right	3
MouthFrown_L	Mouth frown left	19
MouthFrown_R	Mouth frown right	1
MouthSmile_L	Mouth Smile left	20
MouthSmile_R	Mouth Smile right	4
MouthDimple_L	Mouth Dimple left	18
MouthDimple_R	Mouth Dimple right	13
LipsStretch_L	Lips stretch left	NA
LipsStretch_R	Lips stretch right	NA
LipsUpperClose	Lips upper close	27,5,10
LipsLowerClose	Lips lower close	26,16,12
LipsUpperUp	Lips upper up	5
LipsLowerDown	Lips lower down	16
LipsLowerOpen	Lips lower open	16
LipsFunnel	Lips funnel	NA

Table 4.1 - *Continued*

LipsPucker	Lips Pucker	NA
ChinLowerRaise	Chin lower raise	NA
ChinUpperRaise	Chin Upper raise	NA
Sneer	Sneer	9
Puff	Puff	NA
CheekSquint_L	Cheek Squint left	NA
CheekSquint_R	Cheek Squint right	NA

Most of the blendshape variables in the table can be mapped directly to a corresponding servo motor of the PKD android. These variables are scaled linearly to fit the PKD servo motor range. For direct mapping, the scaling of the blendshape variables takes the form

$$P_y = mB_x + c \tag{4.29}$$

Some servo actuators are mapped inversely to the blendshape variable. In that case we can use

$$P_y = m(1 - B_x) + c. \tag{4.30}$$

where Py is the servo position of the motor, m is the scaling factor and c is the offset that is adjusted to get the neutral position.

The head rotations are transmitted from the faceshift using the Quaternion representation. This is converted to roll, pitch and yaw using the following formula.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} atan2(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ \arcsin\left(2(q_0q_2 - q_3q_1)\right) \\ atan2(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}$$

(4.31)

$[q0\ q1\ q2\ q3]^T$ is the unit quaternion transmitted by the faceshift software. The roll, pitch and yaw are further linearly scaled to map the range of motions of the PKD android neck servo motors. These motor values are then transmitted to the Pololu servo controller through serial port using RS 232 protocol. The result of this is that the PKD android copies the neck motions and the facial expressions of the human in real-time. The pictures below show various poses and expressions imitated by the robot in real-time.

Figure 4.10 PKD copying the tilt motion

Figure 4.11 PKD copying the yaw motion

Figure 4.12 PKD copying the smile expression

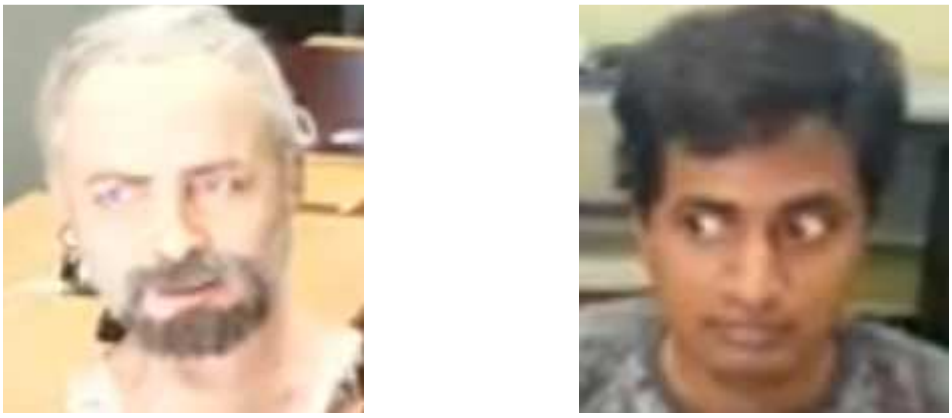


Figure 4.13 PKD copying the eye left motion

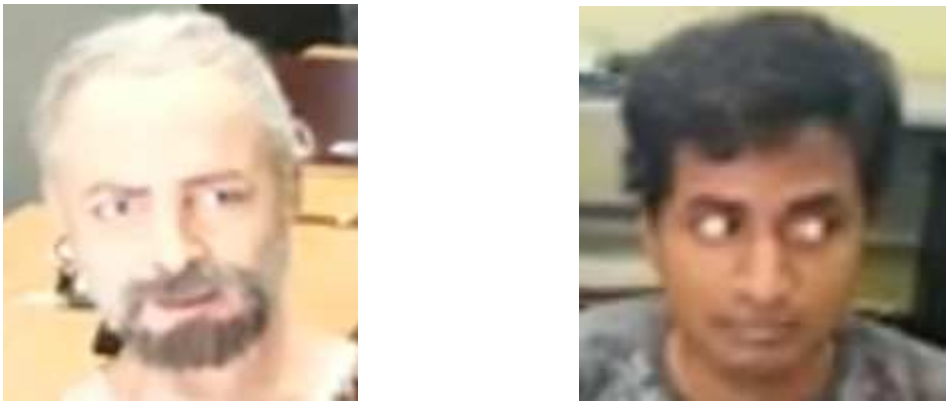


Figure 4.14 PKD copying the eye right motion

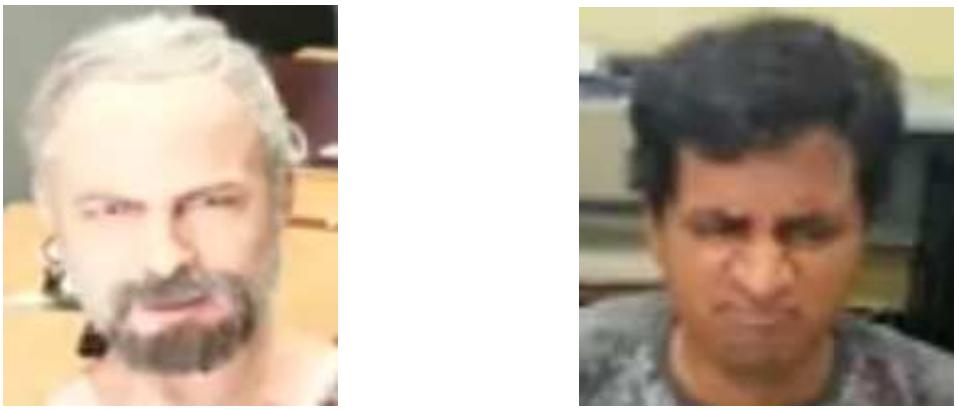


Figure 4.15 PKD copying the sneer expression

In the second method we use an automatic way to map the 48 blend shape variables to the PKD servo motors using a neural network-based learning method. The set up to train the neural network is shown in the fig 4.22. Thousands of random servo values are given to the servo motors of the PKD robot and the corresponding 47 different facial features are recorded using the software Faceshift.

Figure 4.16 to 4.19 shows the PKD command value and the servo values for the jaw, smile left and smile left movements. We can see that the movements are in synchronous with the values obtained from the faceshift. Also we can see the inverse mapping between the smile left faceshift values and the servo motor values.

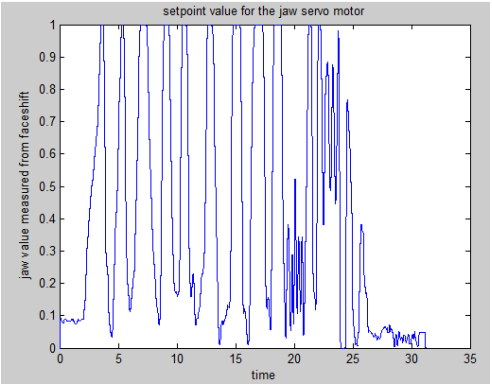


Figure 4.16 Jaw set points vs. time

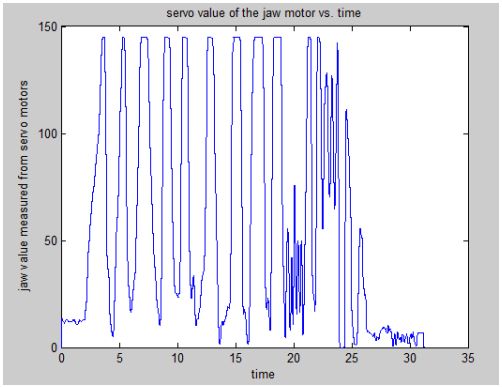


Figure 4.17 Jaw servo values vs. time

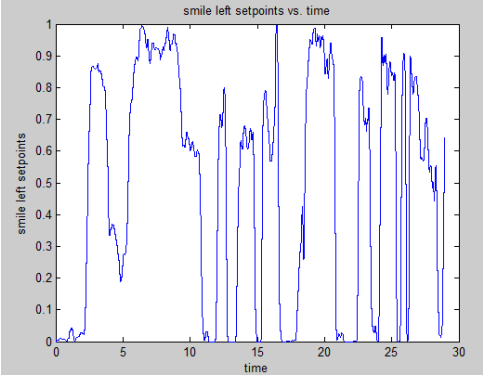


Figure 4.18 Smile left set points vs. time

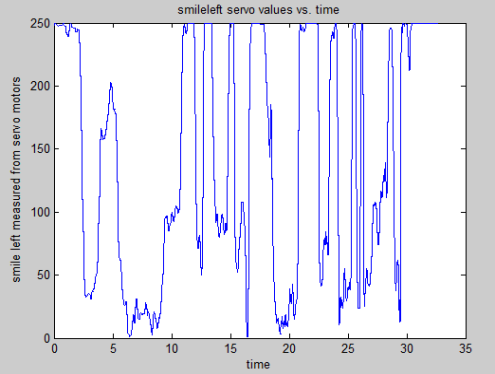


Figure 4.19 Smile left servo values vs. time

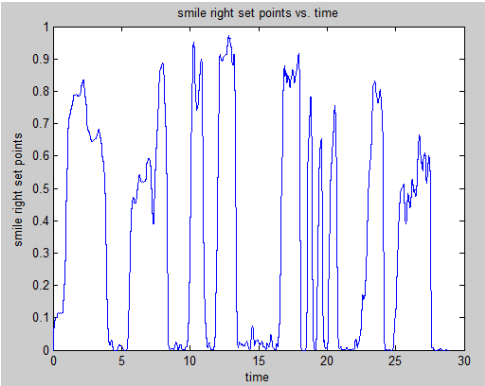


Figure 4.20 Smile right set points vs. time

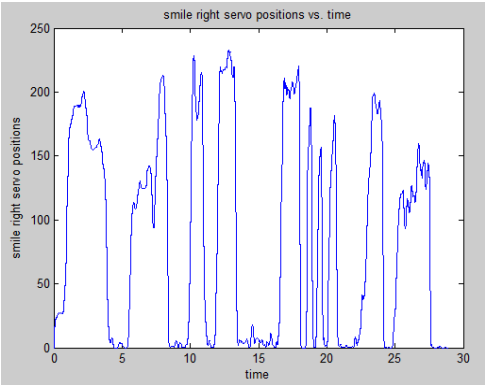


Figure 4.21 Smile right servo values vs. time

4.6.2 Non Linear Neural Network Mapping

In the second method we use an automatic way to map the 48 blend shape variables to the PKD servo motors using a neural network-based learning method. The set up to train the neural network is shown in the fig 4.22. Thousands of random servo values are given to the servo motors of the PKD robot and the corresponding 47 different facial features are recorded using the software faceshift.

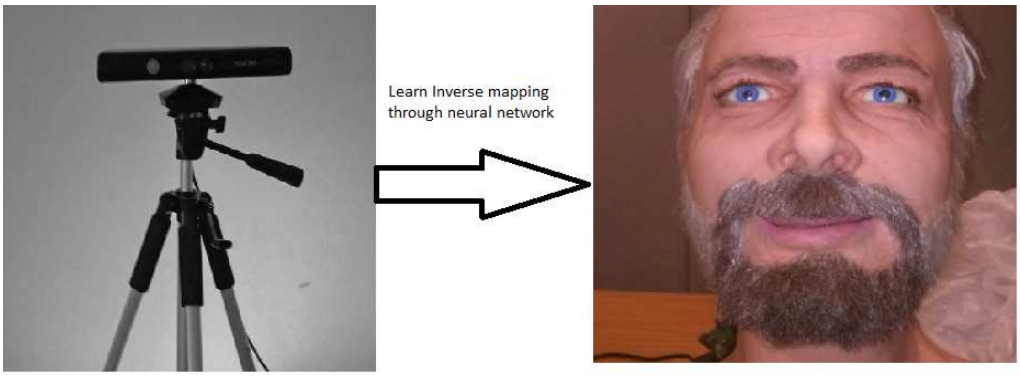


Figure 4.22 Neural Network training setup

The fig 4.23 describes the set up for collecting the minimum and maximum values of the 48 facial features obtained through faceshift software. These minimum and maximum values are used to generate the transformation matrix that will be used to transform 48 facial feature values of the user to that of the PKD and then we use the neural network as an inverse mapping to find the servo values for the corresponding facial feature values.

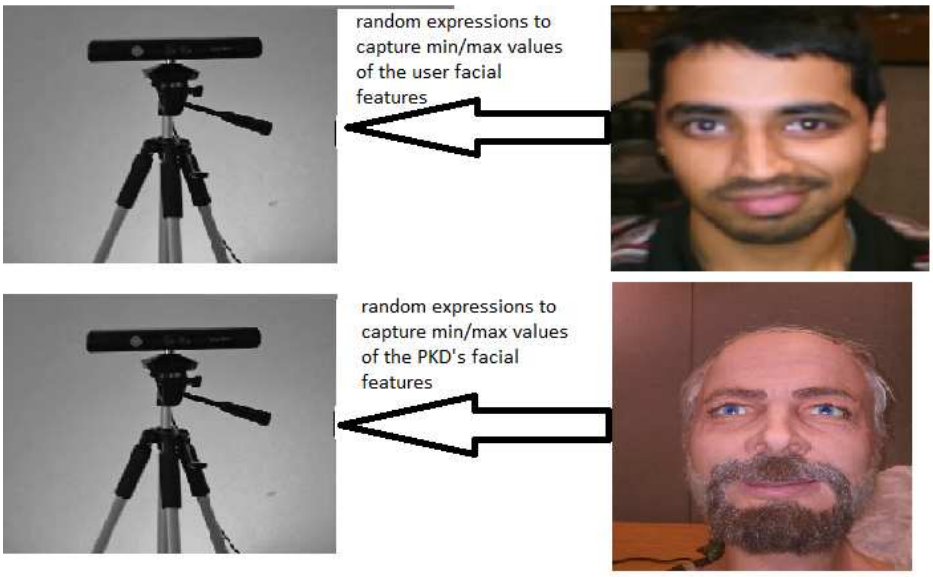


Figure 4.23 Set up to find the transformation matrix between the user and the PKD

The fig 4.24 describes the set up for the real-time facial expressions mimicking by the robot and the user. The 48 facial feature values extracted from the users face using the faceshift software is transformed to that of the PKD's facial features using the transformation

matrix and finally neural network mapping is used to get the servo values of the PKD and all of this is done in a loop at approximately 10 Hz the expressions is mimicked in real-time.

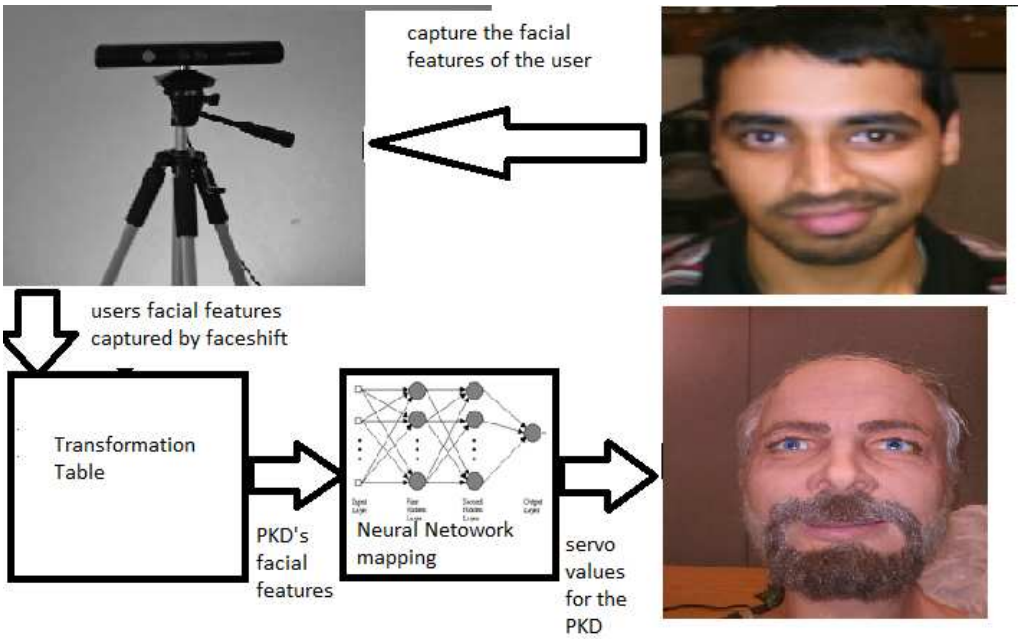
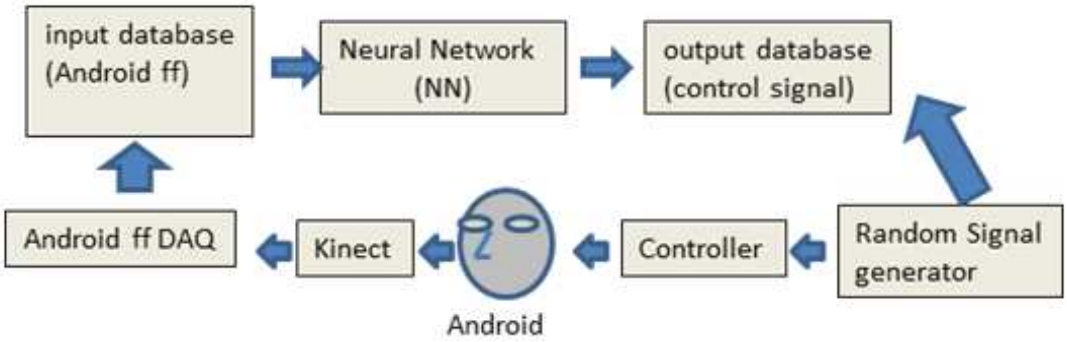


Figure 4.24 Set up for the Real-time facial expressions mimicking

4.6.2.1 Algorithm

Our proposed method employs Neural Network (NN) to learn the mapping between the Android facial expressions and the servo movements. In order to collect the data for NN training, Kinect captures the facial feature of PKD while it generates a series of random servo movements as shown in Figure 4.26



.Figure 4.25 set up for dataset generation for neural network

In order to produce a mapping between the Human Operator's facial expressions and servo movements, a Transformation Map is employed to convert each Operator's facial features to the desired PKD's facial features. Thereby, the trained Neural Network can operate on the stream of converted PKD's facial feature values and produce the required control signals for the actuators. The block diagram for the process is shown in Figure 4.26:

Figure 4.26 Block diagram including transforming map

A two layer Neural Network is used to learn the mapping between the facial features of the PKD and its servo values that generated the facial expressions. The training data set consists of inputs as the PKD facial features and outputs as the corresponding servo command values. These datasets are collected in thousands from the experiment conducted for the mapping of user and android facial features. The neural network is trained with the datasets using the back propagation algorithm. Once the neural network is successfully trained with the datasets we can use it to generate set points for the servo values by giving a desired PKD facial feature points.

Finally we combine the neural network mapping and the transformation mapping described earlier to perform real-time facial expressions imitation. The user's facial expression

features will be captured by faceshift software using Kinect and we transform the user's facial features with the PKD's facial features using the transformation mapping. Finally we give the PKD facial features as input to the neural network to get the servo positions that produces the same facial expressions. We then use these servo positions as commands to the servo controller that uses a PID control to set the servos to the desired positions.

4.6.2.2 Results

At first we tried to train the neural network as one single network with the servo motors of the mouth and the eye brows as output and the corresponding Kinect facial features pertaining to the mouth and eyebrows as the input. After training with the datasets, we found that the eye brows were also moving a little when only mouth was moving. We attributed this behavior with the mechanical failure and degradation of the actuator behavior while generating thousands of datasets. We therefore used a separate neural network for the mouth and another neural network for the eyebrows. For training the mouth the following servo motor positions were used as output: 20, 27, 5, 10, 4, 3, 1, 12, 16, 26, 19, and 18 (numbers are as per figure 4.9) and for eyebrows we used the following servo motor positions: 23, 24, 25, 7 and 6. The corresponding facial features from Faceshift we used as input. The neural network was trained using one hidden layer with 15 neurons using back propagation algorithm in MATLAB.

After the neural network is trained, the experiment is conducted by giving the facial features from the Faceshift as the input. The servo values output of the neural network are then given as set points using the Pololu controller which results in the mapping between the human and the PKD as shown in the figure 4.27

(a)

(b)

Figure 4.27 Non Linear Mouth servos mapping using Neural Networks (a) Jaw motion (b) Smile motion

CHAPTER 5

REAL-TIME CONTROL OF ZENO ROBOT FOR AUTISM THERAPY

5.1 Introduction

The Zeno robot is operated in two modes, interactive mode and the scripted mode. In the interactive mode, the robot will imitate the motion of the human arms and torso in Real-time and in the scripted mode it does preprogrammed gestures like hand wave, tummy rub, and fist bump. The scripted motion of the Zeno is used to conduct experiments with Autism at UNT Health science.

The setup of the system is as shown in the fig 5.1 below. The Microsoft Kinect is used as an interface device to run the interactive mode. The skeleton tracker from NITE middleware is used to track the human joint angles in real-time. The Dell XPS laptop is used to run the computer vision algorithm. OpenNI SDK installed in the Dell laptop will provide the device driver interface for all the depth sensors like Microsoft Kinect, ASUS etc.

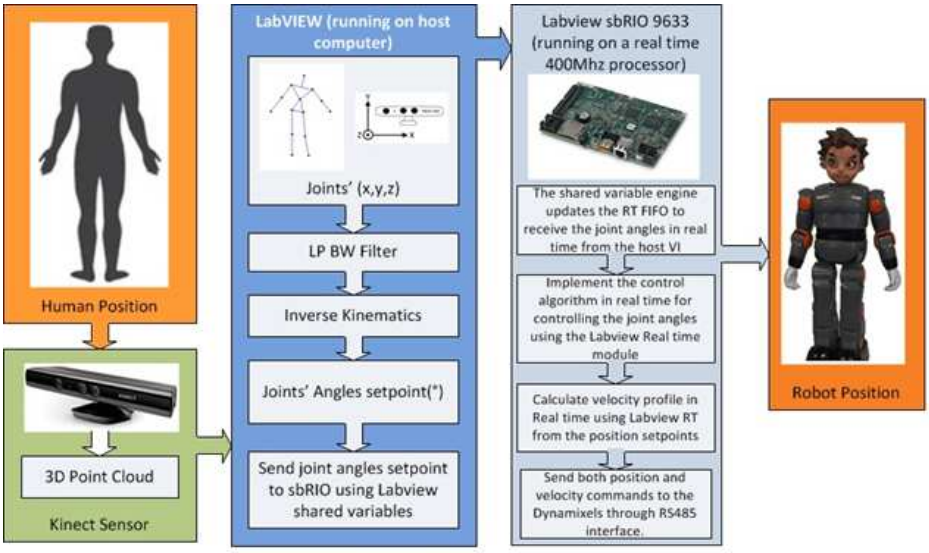


Fig 5.1 Zeno System Architecture

The LABView sbRIO9633 is used to control the real-time motion of the Zeno that does the position and velocity control.

5.2 Zeno Running in the Interactive Mode

The Zeno running in the interactive mode occurs through a series of process. The VI that runs on a desktop based computer will process the data from the Kinect and its front panel is shown in the fig below.

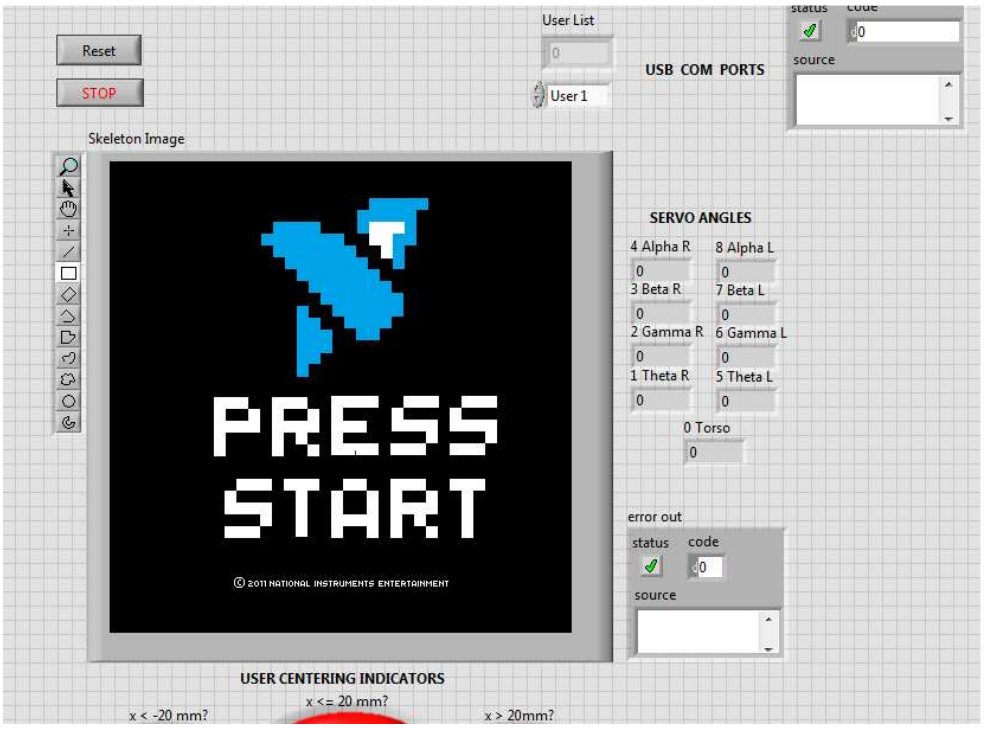


Figure 5.2 Front Panel of interactive mode VI

The user has to stand in a calibration (“psi”) pose before the system could build his skeleton and the quality of the skeleton improves with time. These skeletal coordinates are passed through the low pass filter as shown in the fig 5.3. It retrieves specific joint coordinates from the Kinect data cluster. The data points from the Kinect are pretty noisy and a 2^{nd} order low pass Butterworth filter with frame rate of 25 fps, and cut off frequency of 2 Hz is used to smooth the signals. The filtered joint angles of the human needs to be mapped to that of the robot joint angles. The human has three degrees of freedom at the shoulder, and two the elbow but the

Kinect only captures the one degree of freedom at the elbow. The robot on the other hand has two degrees of freedom at the shoulder and two degrees of freedom at the elbow. This mapping between the human and robot joint angles is done in the inverse kinematics VI (IK) in the fig 5.3.

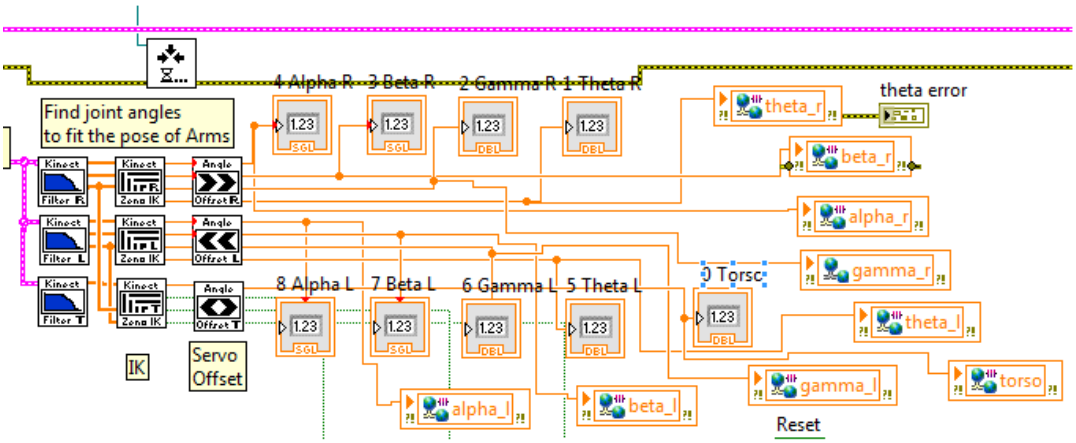


Figure 5.3 Kinematics mapping, filtering of the joint angles

To extract the alpha and beta of Zeno shoulder joint we use the Euler ZYZ parameterization. Therefore alpha and Beta can be extracted as

$$\alpha = Atan2\left(\frac{-z}{-y}\right)$$

$$\beta = cos^{-1}\left(\frac{x}{\sqrt{x^2+y^2+z^2}}\right) \tag{5.1}$$

In similar ways the gamma and theta of the elbow joint can be extracted by considering the Euler ZYZ parameterization at the elbow joint.

The robot joint angles alpha, beta, gamma and theta of both arms and the torso movements are then updated to the LabVIEW shared variable. These shared variables are read in the real-time joint control VI running in the sbRIO through the LabVIEW Real-time shared variable engine. The front panel of the VI running the sbRIO is shown in the fig 5.4.

The function of sbRIO is to control the motors of the robot in real-time. This is done by sending the position and velocity commands to the motors every 20 Hz

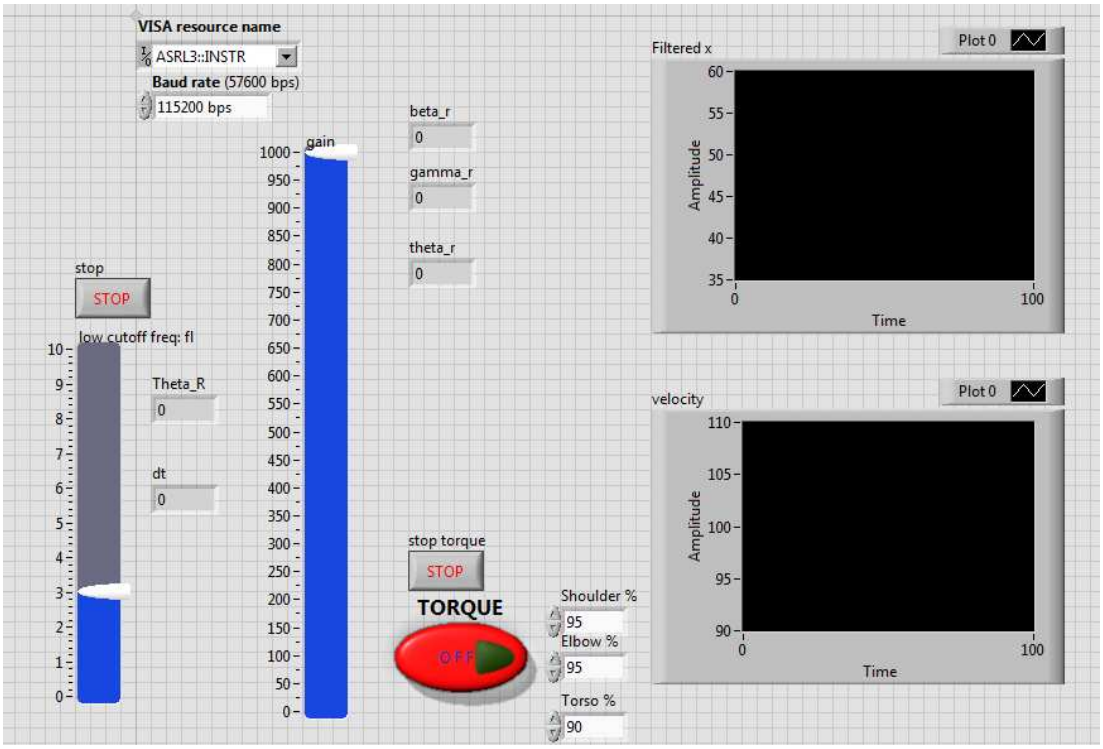


Figure 5.4 Front Panel of real-time control VI of sbRIO

The baud rate in the front panel is the rate of communication between the sbRIO and the Dynamixel motors of the robot. This is selected to be 115200bps for the 2 Wire RS485 interface. This is the maximum baud rate that the sbRIO supports. The visa resource should be selected as ASRL3 in the drop down menu as this indicates we are choosing the RS 485 channel. The torque Stop can be used to stop the torque in the motors. The two plots show the position and velocity of the theta value on an arm that is sent to the Dynamixel motors to control the robot. The Real-time VI is scheduled to run at every 50ms as shown in the figure 5.5. This is as low as it can be scheduled due to the maximum limitation of the baud rate.

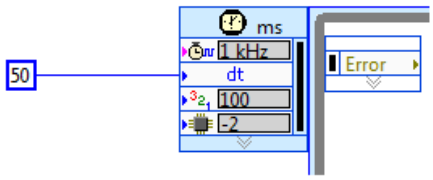


Figure 5.5 scheduling rate of the VI

The alpha, beta, gamma and theta values are received from the VI that runs on the computer extracting the human joint angles through LabVIEW shared variable engine. The position variables are then low pass filtered using the Butterworth filter. The velocity is calculated using

$$velocity = \frac{P_t - P_{t-1}}{t - t_{-1}}.$$

(5.2)

Where,

The numerator is the difference in the position variable between the current time t and the time t-1. The denominator is the time elapsed between the last update of the position variable and the current time. This time is calculated in the elapsed timer below. The velocity and the position values thus generated in this VI are sent as commands to the RX28 Dynamixel motors through RS 485 communication channel in the sbRIO.

5.3 Zeno Running in the Scripted Mode

The gestures that are used for the scripted mode are the hand wave, tummy rub and the fist- bump for both of the arms of the robot. These gestures are pre-recorded in files and are played back in various combinations. The VI that was used to control the Zeno robot in the interactive mode was used to recording the various gestures. The difference is that now the joint angles are not channeled to the Dynamixel motors, instead they are recorded and stored in a file as a sequence of position commands. Another VI was developed to read the position values stored in the measurement files and send those position values to the Dynamixel motors through the RS 485 channel. Now we can also control the speed of the gesture. In the front panel, the Zeno motion speed control controls the frequency at which the position values are read from the measurement file. By changing this we can control the desired speed for a particular gesture. For the regular ASD trial, all motions are played three times consecutively. This can be selected using the playback motion dropdown menu shown in the front panel below. After selecting the appropriate files that contain the recorded motions on the left of front

panel and the appropriate serial port in the VISA Arms, the scripted motion is ready to be played.

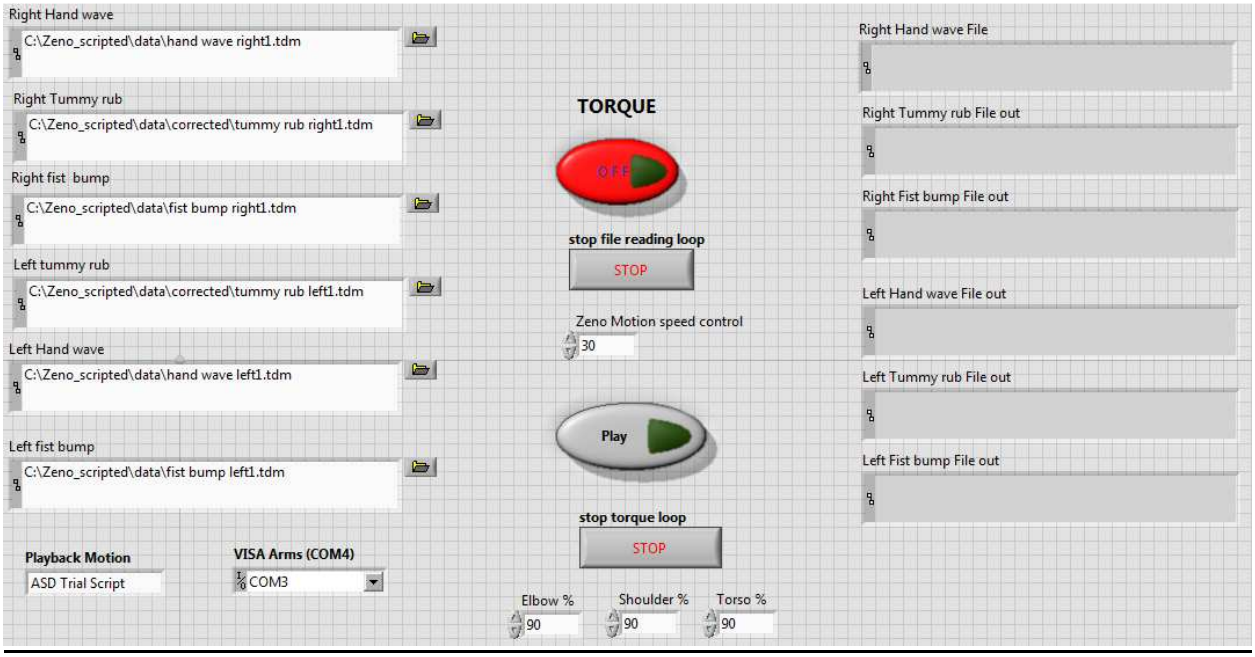


Figure 5.6 Front Panel of the scripted mode

5.3.1 Data Collection and Processing of Human Subjects

Figure 5.7 shows the markers placed on Zeno and the child used for collecting the motion data of the child and the robot. The data collection is performed using the motion capture system. These experiments are carried out at UNT Health Science center where the therapists ask the child to imitate the motions of the robot so that we can capture the data of both the child and the robot. We use Dynamic Time Warping as a tool used for comparing the motions of the Zeno and the child [49]. The result of this method is that we get a number indicating how similar the motion of the Zeno and the child is.

Figure 5.7 Markers on Zeno and the child for data collection

5.4 Results and Conclusion

5.4.1 Zeno Motion Results using sbRIO 9633

The figures below show the plot of the commanded servo position value vs. the actual motor encoding value. It can be seen that the Dynamixel motors at the joints follows closely the command signal and due to the real-time control that is possible with the sbRIO, we can see that the Dynamixel motors have smooth changes in the position commands even when the desired position is changing fast. This is possible since we are giving the velocity commands along with the position commands to the Dynamixel motors.

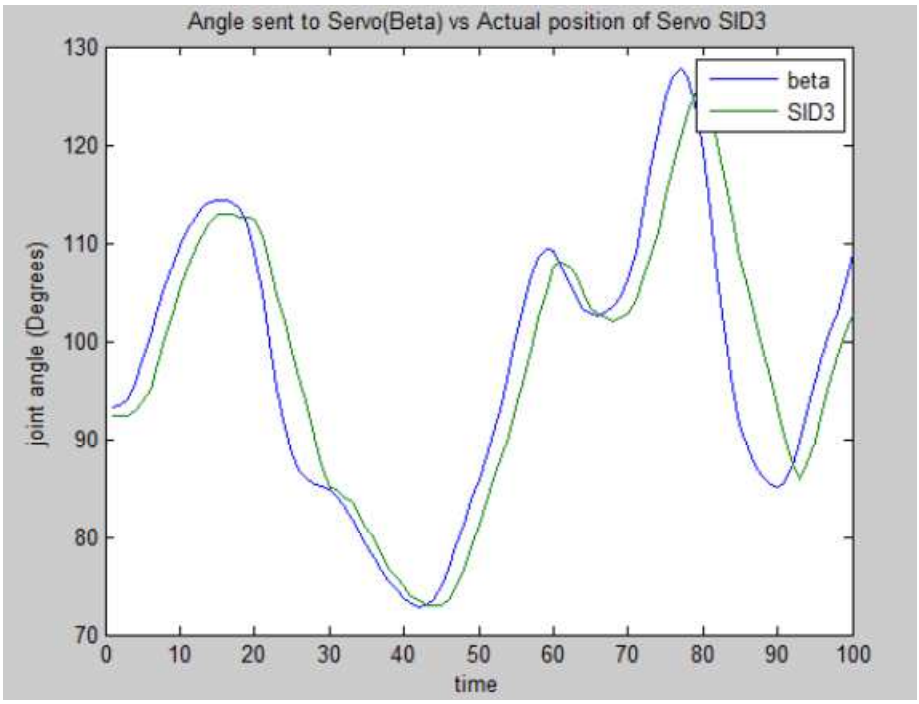


Figure 5.8 Angle sent to Servo(Beta) vs Actual position of Servo SID3

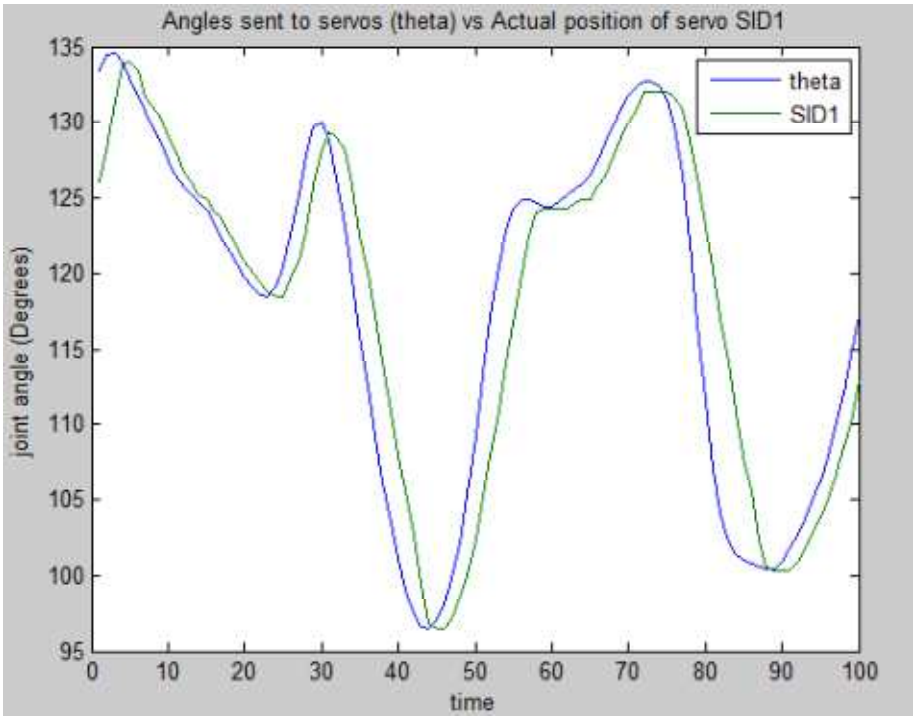


Figure 5.9 Angle sent to Servo(theta) vs Actual position of Servo SID1

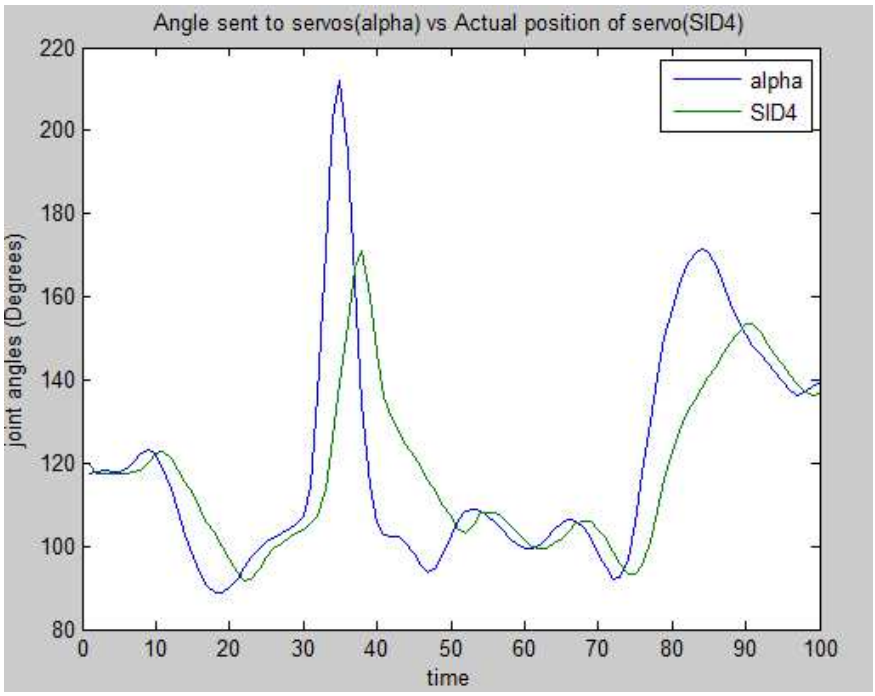


Figure 5.10 Angle sent to Servo(alpha) vs. Actual position of Servo SID4

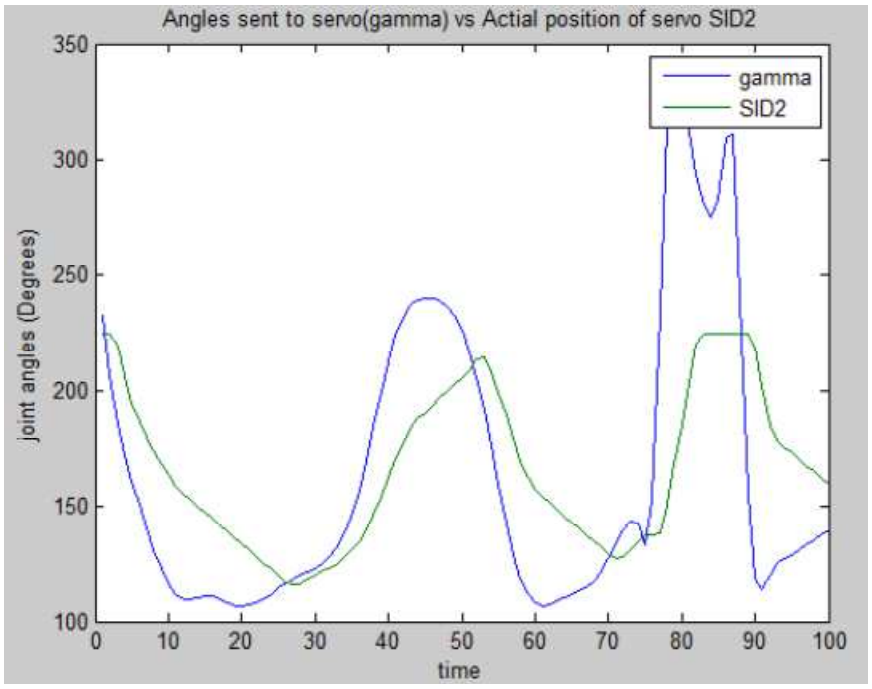


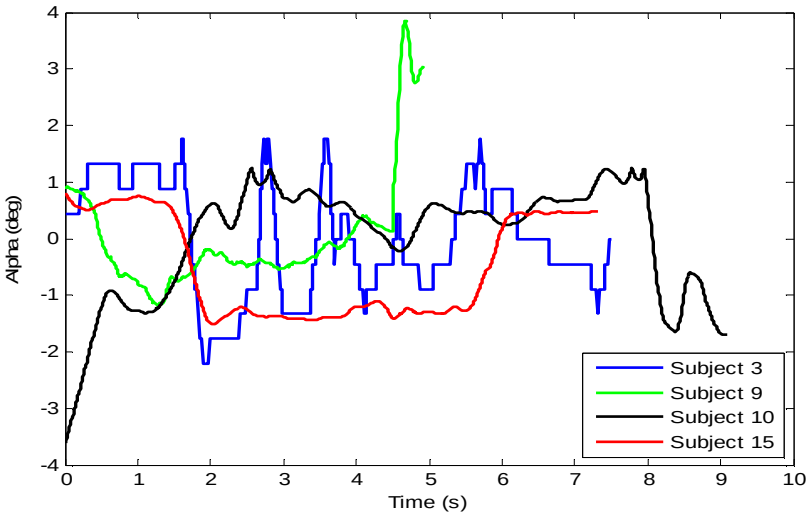
Figure 5.11 Angle sent to Servo(gamma) vs Actual position of Servo SID2

The advantages of using the sbRIO real-time controller for controlling the robot is as follows

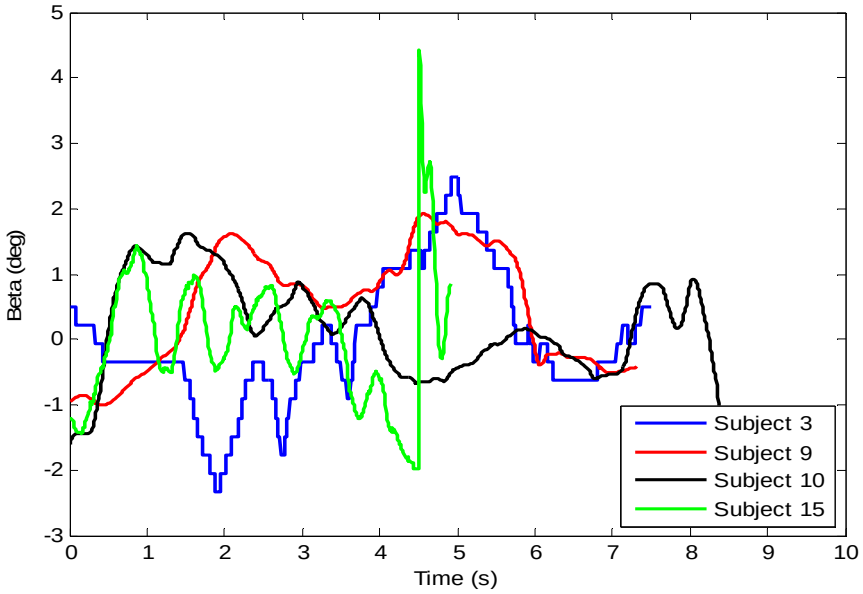
- Smooth motion with velocity commands
- Advanced control algorithm like computed torque control can be implemented
- We can do real-time force control

5.4.2 Results of the Data Collection process

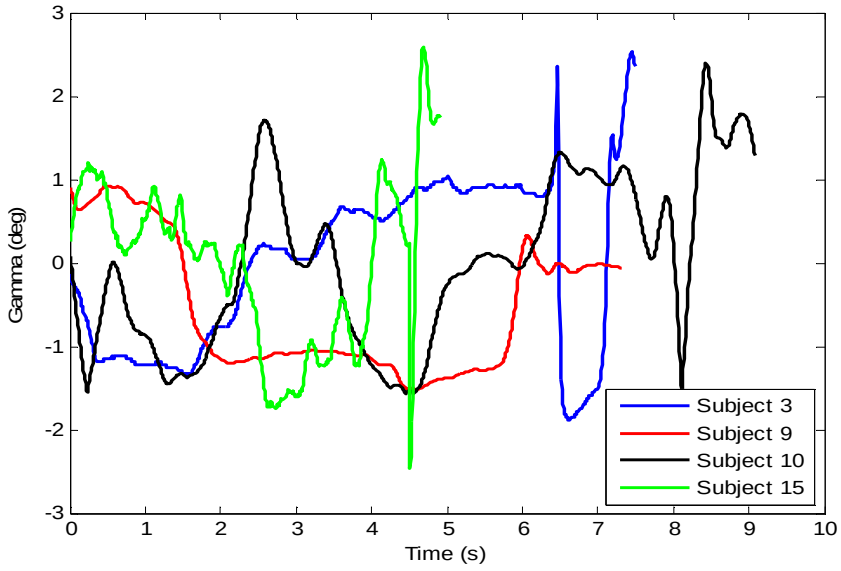
A set of clinical experiments with several children were performed to test the robot, capture the motion, and to compare the motion data using DTW. During experiments, children were directed to follow the hand gestures of the robot, such as wave with hands, tummy rub, fist bump, etc. Each motion was performed three times. The best one was picked for analysis. Representative joint angle trajectories for each DOF is shown in the fig 5.11



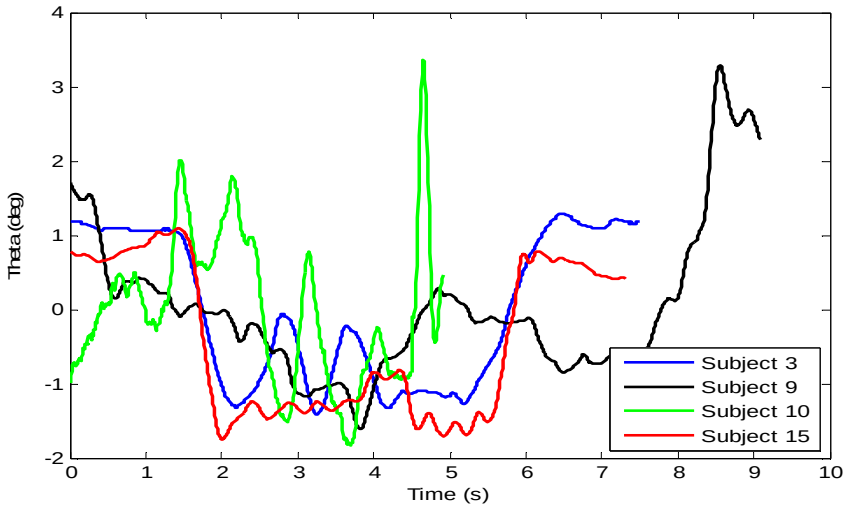
(a)


Beta (deg)
Time (s)
Subject 3
Subject 9
Subject 10
Subject 15

(b)


Gamma (deg)
Time (s)
Subject 3
Subject 9
Subject 10
Subject 15

(c)



(d)

Figure 5.12 (a), (b), (c), (d) Normalized Joint angles

Wave gestures are compared for the subjects3, 4, 6, 7, 9, 10 and 15. The DTW distance between arm angles of Subjects and Zeno are calculated and is tabulated as shown in the table 5.1 below:

Table 5.1 DTW distance between arm angles Subject and Zeno

Subject #	Alpha	Beta	Gamma	Theta	ASD/Control
Subject 3	0.6807	0.6807	0.6807	0.1117	Control
Subject 4	0.8392	0.8392	0.8392	0.2931	ASD
Subject 6	0.803	0.803	0.803	0.2756	ASD
Subject 7	0.3141	0.3141	0.3141	0.117	ASD
Subject 9	0.7466	0.4806	0.4968	0.7506	ASD
Subject 10	0.9663	0.857	0.6414	0.2643	Control
Subject 15	0.3837	0.3463	0.3267	0.3171	ASD

CHAPTER 6

ATLAS ROBOT GETTING UP FROM ITS PRONE POSITION

6.1 Introduction

This project was done as a part of DARPA Robotics Challenge. In this project a script was developed that will make the robot get up when it falls down. There are several pioneering researches concerning the getting up motions of small-size humanoid robots. “Hanzou”, developed is the first humanoid robot that can get up by itself (Inaba et al. 1995a)[59]. It can also continue to walk even after it tips over. Hanzou has a height of 30 cm, a weight of 2 kg, and 16 degrees of freedom (DOF)[60]. When it tips over and is lying on its back, it can roll over by swinging its legs. The motion of getting up is segmented into a sequence of contact states between the robot and the floor, but each transition between consecutive states is realized by a playback of a fixed motion pattern. When the Atlas robot falls down, the trajectories are generated such that the robot goes through various contact states in the process of getting up. Contact states are the positions the robot has to go through in order to get up. These trajectories are then filtered using cubic spline interpolation to generate the position and velocity set point trajectories that complies with the minimum and maximum position and velocity constraints of each joint of the atlas robot. Finally, when the script is run the robot gets up on its feet successfully.

6.2 Details of the Atlas Robot Getting up from its Prone Position

Getting up is a motion that changes the contact state between the robot and the floor from a state at which it’s front or back contacts the floor to another state at which its feet contact the floor. The static balance should be kept during a transition from one contact state to another. The transitions can be realized by controlling the relationship between the projection of the Center of Mass (CoM) onto the floor and the supporting polygon of the robot.

The support polygon is defined as the polygon formed by the contact points of the robot with the ground. The robot pose is considered to be stable if the projection of the CoM of the robot lies within this support polygon. The robot has to go through a sequence of several contact states in order to stand up as shown in the fig 6.1. Getting up task has been done successfully done and there is good number of papers published with various robots like the HRP2.

Once the contact states are defined, we need to now interpolate the start and end states of the joint angles to generate the trajectories. For this we shall use the cubic spline interpolation method. A cubic polynomial has 4 coefficients, and hence may be used to satisfy both position and velocity constraints at the initial and final positions.

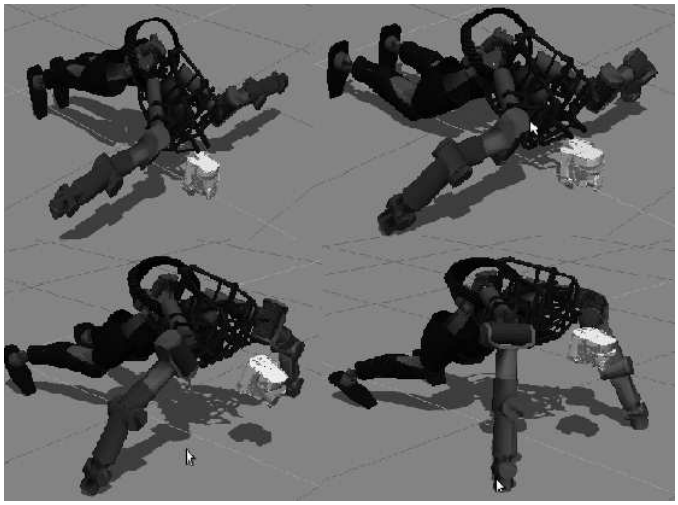


Figure 6.1 First four contact states while getting up from its prone position

For joint variable qi, the constraints are:

$$q_i(t_0) = q_0$$

$$\dot{q}_i(t_0) = q_0'$$

$$q_i(t_f) = q_1$$

$$\dot{q}_i(t_f) = q_i'$$

where t_0 is the starting time,

t_f is the ending time,

q_0 and q_0' are the specified initial position and velocity

q_1 and q_1' are the specified final position and velocity

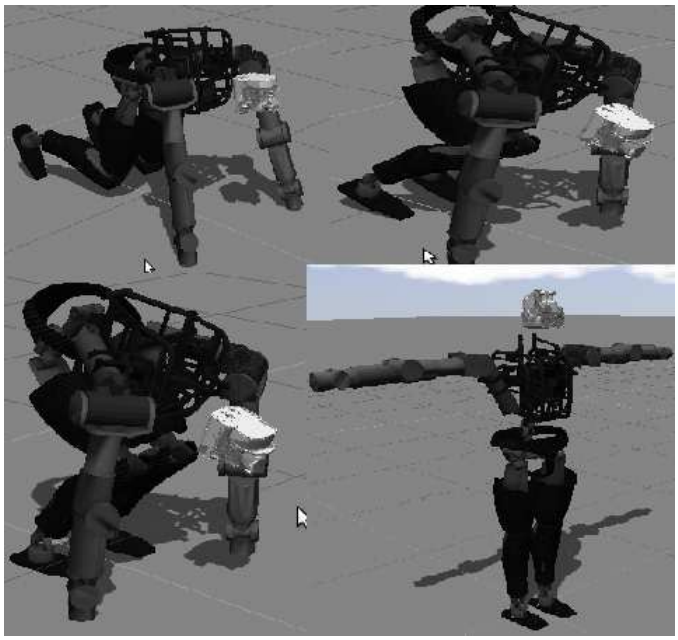


Figure 6.2 Last four contact states of the robot while getting up from the prone position

The cubic polynomial for joint positions and its derivative for joint velocity are

$$q_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{q_i}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

Then we have $a0$ and $a1$ from the initial conditions

$$q_i(0) = a_0 = q_0$$

$$\dot{q_0}(0) = a_1 = q_0'$$

Next we find a_2 and a_3 from the final conditions. Substituting for a_0 and a_1 evaluated at $t_f$we get

$$q_i(t_f) = q_0 + q_0' t_f + a_2 t_f^2 + a_3 t_f^3 = q_1$$

$$\dot{q_i}(t_f) = q_0' + 2a_2 t_f + 3a_3 t_f^2 = q_1'$$

$$3q_1 - t_f q_1' = 3q_0 + 3q_0' t_f + 3a_2 t_f^2 + 3a_3 t_f^3 - q_0' t_f - 2a_2 t_f^2 - 3a_3 t_f^3$$

$$= 3q_0 + 2q_0' t_f + a_2 t_f^2$$

Hence,

$$a_2 = \frac{3(q_1 - q_0) - (2q_0' + q_1')t_f}{t_f^2}$$

$$a_3 = \frac{2(q_0 - q_1) + (q_0' + q_1')t_f}{t_f^3}$$

The method described above is used by trajectory filter package in ROS that will generate the trajectories by taking the position and velocity limits of each joint as we describe in joint_limits.yaml file. This yaml file is populated from the joint limit values from the URDF file. The starting and the ending position described for just one joint result in the trajectory as shown in the fig 6.3 below.

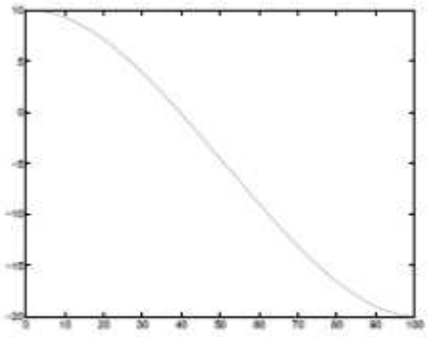


Figure 6.3 Trajectory smoothing using cubic splines.

After generating the trajectories by interpolating the contact states, the trajectories are then published to the joint controllers in a sequential manner which makes the robot get up in a smooth way.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis, we have addressed some of the challenging problems involving real-time natural human robot interaction on various robotic platforms like the social robot Zeno, the android PKD and the Atlas robot in Gazebo simulation. The focus has been to develop natural interactions with humans in robots while having a real-time implementation

Starting with a simple Pan and tilt control of the robot that tracks faces with two degrees of freedom we developed algorithms that can be used for real-time face tracking in androids, real-time imitation of human motions by humanoid robots and real-time mimicking of facial expressions by the facially expressive robots. The motivation for this is the need for the robots to be more acceptable where humans work together with robots and the need of social robots in assisting people with medical condition like Autism therapy.

7.1.1 Real-Time Face Tracking by Android PKD

In this part of the thesis we are interested in androids having natural interaction with humans by making eye contact with humans and moving the neck and eyes like the way a human moves. The motivation behind this is to impart social skills to robots while interacting with humans so that they don't appear robot like. This will be important for therapy sessions with Autistic children. A novel algorithm was developed to distribute the motions between the neck and the eyes by solving the redundancy in the motions using separate PID control and formulating the error conditions for them. This enables the children with Autism make a connection with the android so that the robot can now teach social skills to them.

7.1.2 Real-Time Imitation of Motions and Facial Expressions by Androids

In this part of the thesis we are able to extract the human arms, legs, torso motions and the facial expressions of the humans and map it to the robots to its actuator space. One of the factors that influenced to develop the imitation of motions by robots was to be able to teach people with medical condition like Autism through the robots. Another motivation was to be able to tele-operate robots in a hazardous environment where the humans cannot be present and operate in environments where the Fukushima nuclear disaster took place. A novel way to extract the human joint angles through Euler and Quaternion representations were tested on Atlas robot. We also developed the mapping between the facial expressions of the humans to the androids and tested it on the PKD avatar. The tracking by the robots is promising to be used on such applications as discussed.

7.1.3 Real-Time Arms and Torso Control of the Zeno Robot for Autism Therapy

In this part of the thesis we are able to isolate the low level real-time control of the robot and high level Kinect sensor data extraction and kinematics conversion. The real-time controller sbRIO was used for low level motion control and the Dell XPS was used for high level design. The result is that we are able to control the motions of the robot in fast yet smooth manner by performing velocity control. The scripted motion of the robot was used for Autism therapy sessions held at UNT Health science. These experiments were conducted to study the motions of the Zeno robot and the child and to compare the similarity of the motions. Experiments show an improvement in the motions of the child over a period of time when we used the same repeated gestures by the robot.

7.2 Future work

Further research in natural Human robot interaction can include physical interaction by the use of impedance control in robots. Force sensor arrays can be mounted on the arms of the robot that uses motors in which we can control torque. Complex control algorithms like

computed torque control can be implemented in real-time with this set up so that the robots can interact with the humans physically. This research may be useful for making the robots operate safer in the presence of humans and may be used to interact with Autistic kids for developing their motor skills.

The Androids like the PKD gets a lot of media attention as the algorithms that we developed at NGS laboratory makes the robot look as though it has life into it. This robot along with Zeno was exhibited at the Humanoids2013 [61] conference at Atlanta where it attracted a lot of professors and researchers in Robotics. One of the projects that David Hanson is working along with Dimitry Itskov is the research involved in uploading the mind to the android robots in order to extend life [62]. This needs a lot of research progress in the area of Artificial intelligence. Future research work with Androids like PKD can involve improving the response of the face tracking algorithm even when the face detection fails in some cases. It can also be made robust by rejecting wrong faces detected by the face tracker. Optic flow can be used to increase the speed of face tracking of pixels as it is computationally less expensive.

APPENDIX A

PKD HARDWARE AND SERVO MOTOR DETAILS

Figure A.1 PKD skull without the skin

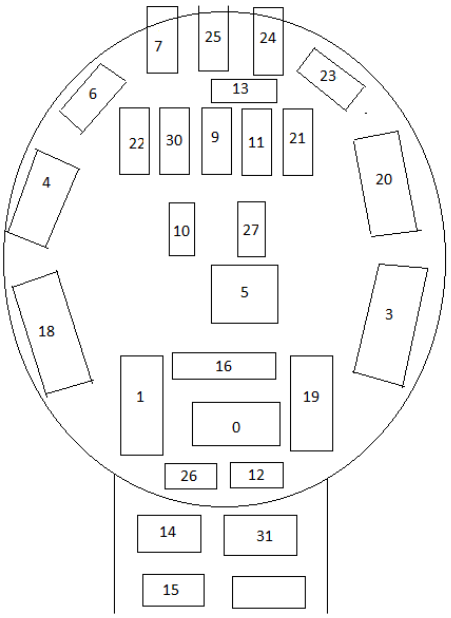


Figure A.2 PKD servo motor placement

Table A.1 PKD servo motors torque and Manufacturer details

.

PKD servo positions	Servo motor used	Maximum torque
18,3	SAV-SC1268MG-H	208 oz-in
4,20,28,14,31,0,5,1,19	Hitec HSR-5990TG	333.3oz-in
26,12,7,25,24,11,21,23,13, 22,30,9	Hitec HS-82MG	2.8 kg-cm
2	HS-805BB	343 oz-in

The figure A.3 shows the overall Block diagram of the connections made to the PKD. The servo motors are controlled with two different 24 channel Pololu servo controller. One servo controller is used for controlling the neck and eye servo motors and the other is used to control the facial expressions servo motors. The idea behind this isolation was to control the neck eye coordination and the facial expressions separately using two different software operating on two independent serial ports. The mode of communication used to talk to the Pololu is the RS232 communication. The commands used to talk to the Pololu are the Mini SSC format.

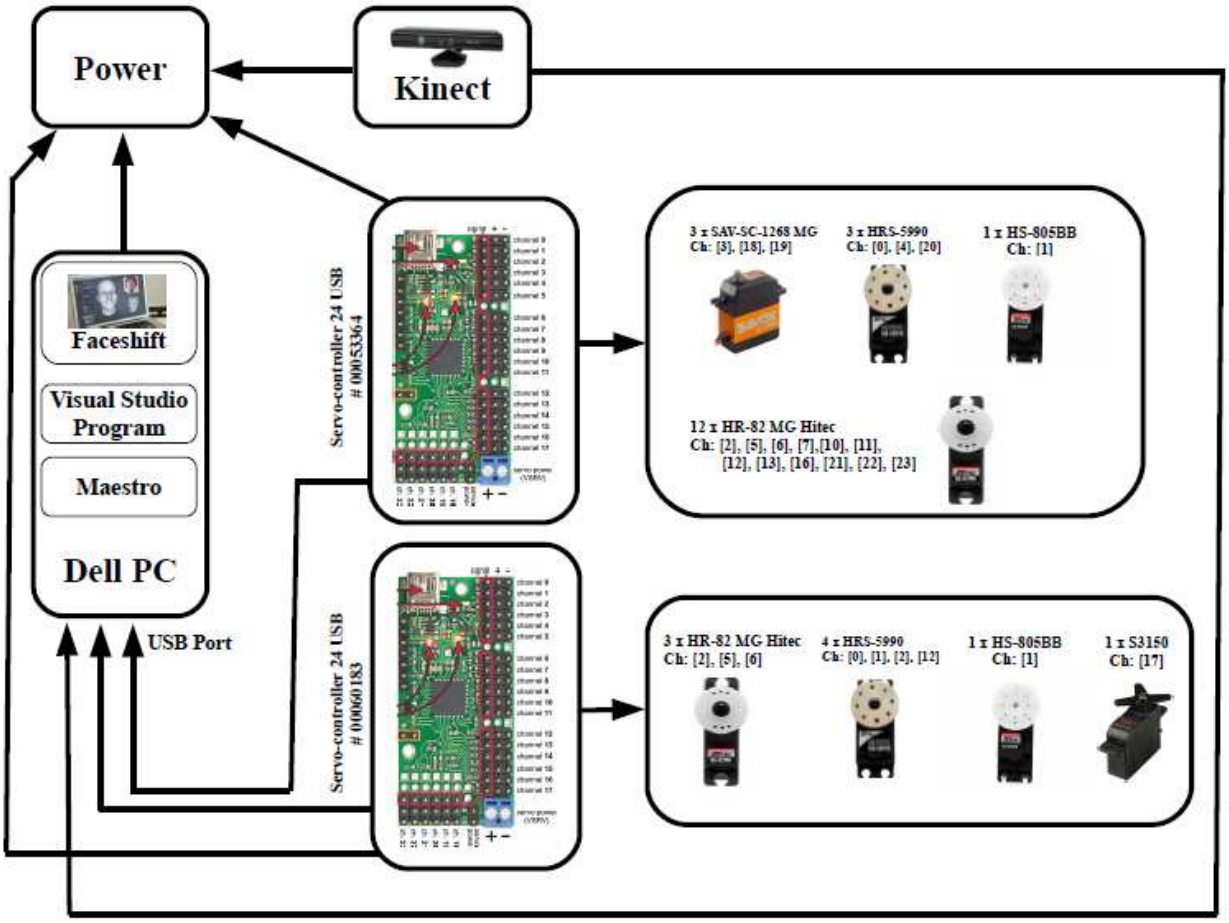


Figure A.3 Block diagram of the PKD Electrical Connections

APPENDIX B

SOFTWARE ARCHITECTURE OF IBVS AND PBVS BASED FACE TRACKING

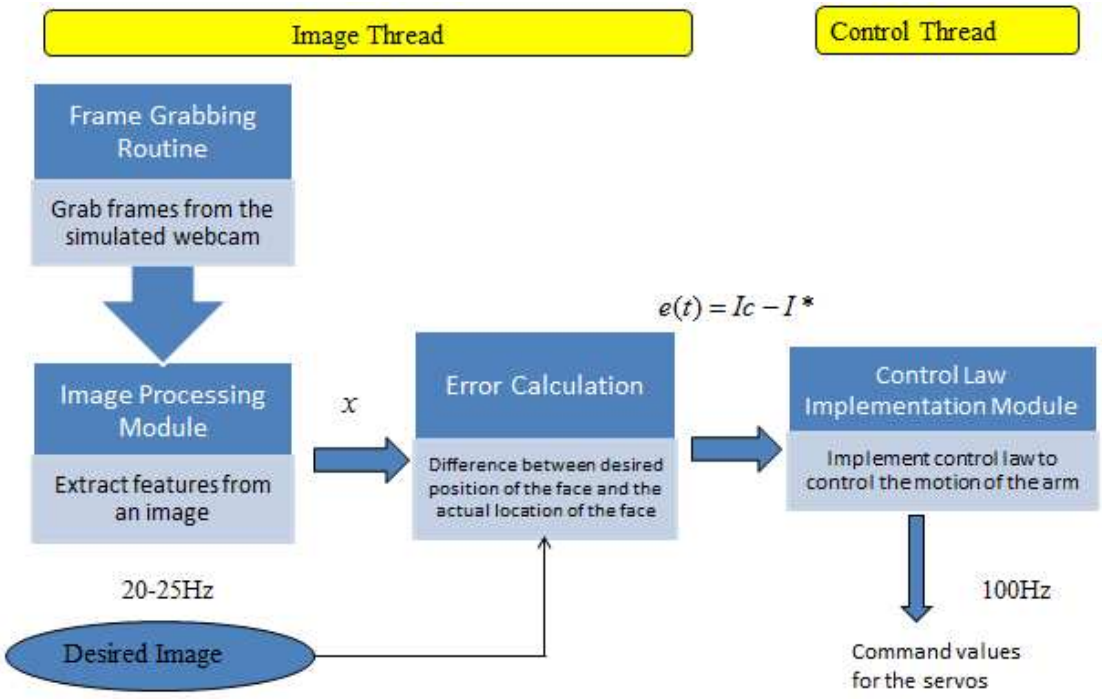


Figure B.1 Software Architecture of IBVS Pan Tilt Face Tracking

In the Image Based Visual Servoing two threads are created. The Image thread and the Control thread. The image thread grabs the image from the USB webcam and detects the face using OpenCV face tracking Software that is based on Haar Cascade classifier. The code below shows the functions in OpenCV that does the image grabbing and the face detection

```
for(;;)
        {
            IplImage* iplImg = cvQueryFrame( capture );
            frame = iplImg;
            if( frame.empty() )
                break;
            if( iplImg->origin == IPL_ORIGIN_TL )
                frame.copyTo( frameCopy );
            else
                flip( frame, frameCopy, 0 );
```

```
            detectAndDraw( frameCopy, cascade, nestedCascade, scale,
            tryflip );

            if( waitKey( 10 ) >= 0 )
                goto _cleanup_;
        }
```

In the above code the function cvQueryFrame is responsible for grabbing the images and the function detectAndDraw detects the face in the image and draws a circle covering the face. The control thread formulates the error condition and implements the PI controller as shown below

```
errorx = centerx - 302;
errory = centery - 236;


pwm1 = pwm1*Ki + (Kp)*(errorx);
pwm1 = (int)(pwm1 + (double)Kd * (errorx/time));


pwm2 = pwm2*Ki - (Kp)*(errory);
pwm2 = (int)(pwm2 - (double)Kd * (errory/time));
```

In the above code the errorx and errory are the difference between the center position of the face detected by the Image thread and the camera center which is found using the camera calibration function in OpenCV. The control signal to the servo motors pwm1 and pwm2 are calculated by adding the proportional, integral and the derivative component which is then fed as set points to the Pololu servo controller via RS232 communication.

The figure B.2 shows the Software implementation of PBVS based Pan and Tilt Control for Face tracking.

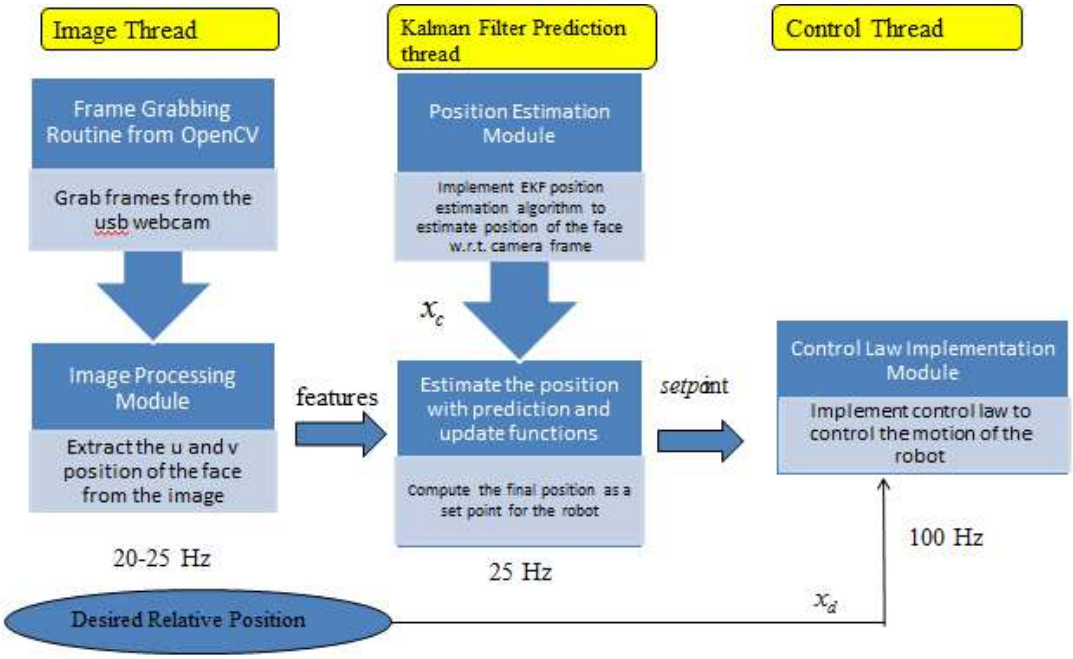


Figure B.2 Software Implementation of PBVS based Face Tracking

The image thread is used to grab images from the camera and detect the face in the image as before. The Image thread is scheduled in 20 Hz because that is about the time it takes for running face detection on the image. The Kalman filter prediction is implemented in a separate thread. In this thread the Predict function is implemented where it predicts the next position in the 3D space based on the constant instant velocity model. Finally the control thread implements a PI control to servo the motor. The following code shows the implementation in the predict thread

```
A[1] = 0.02;
A[1] = 0;
A[15] = 0;
memcpy( kalman->transition_matrix->data.fl, A, sizeof(A));
const CvMat* prediction = cvKalmanPredict( kalman, 0 );
Zc = 649*3.02/rad;
predictX = prediction->data.fl[0];
thetaX = atan(predictX/Zc)*57.295;
```

```
predictY = prediction->data.fl[2];
thetaY = atan(predictY/Zc)*57.295;
```

The cvKalmanPredict is the library function that is called for predicting the next state. The A matrix is updated with the constant velocity values.

APPENDIX C

SOFTWARE ARCHITECTURE OF ATLAS ROBOT TELEOPERATION

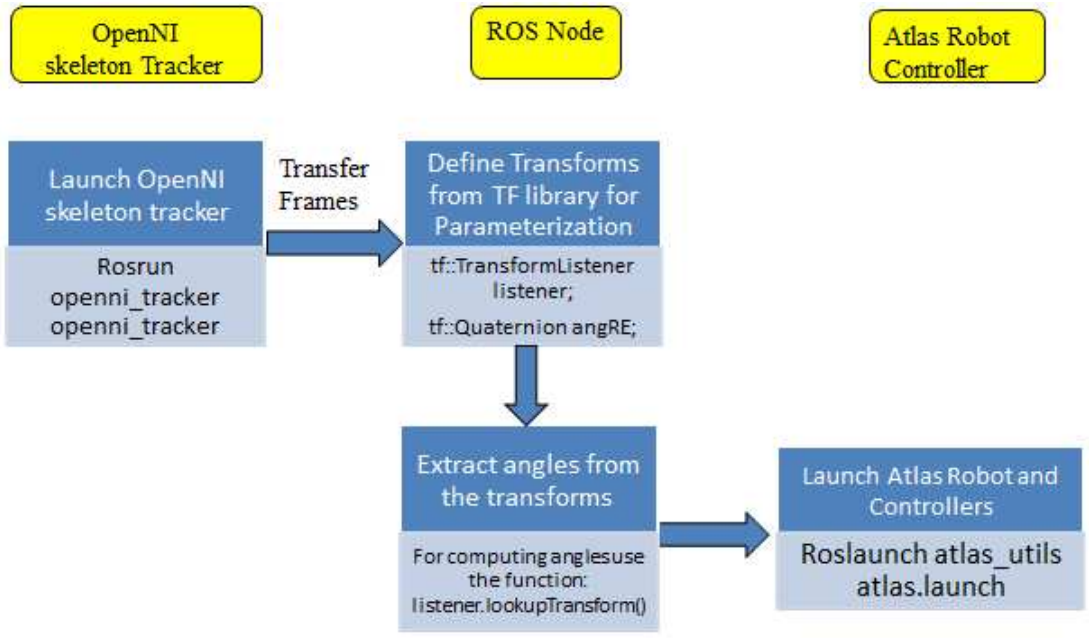


Figure C.1 Software Architecture for Atlas Robot Teleoperation

The figure C.1 shows the Software implementation of the Atlas robot tele operation. First we need to run the skeleton tracker from OpenNI to transmit the frames for all the joints. This is shown under OpenNI skeleton tracker. A ROS node is defined that extracts the joint angles from the transform. The following definition defines the transforms for the Euler angles and the Quaternions.

```
tf::Quaternion angrightknee; // for elbow and knee joints
tf::StampedTransform sh_transform, leftknee_transform, rightknee_transform;// for
//all other joints
```

The following code extracts the angles represented by these transforms

```
listener.lookupTransform("/left_shoulder_1", "/left_elbow_1",
                                        ros::Time(0), el_transform);
sh_transform.getBasis().getRPY(Rs, Ps, Ys); // use for Euler angles
el_transform.getBasis().getRotation(angRE); // use for Quaternions angles
```

The function listener.lookupTransform takes the input of the rostopics ‘/left_shoulder’ and ‘/left_elbow_1’ from the OpenNI skeleton tracker and compute the transform and stores it in el_transform. This is further used in the functions getRPY for getting the roll, pitch and yaw angles from the Euler representation and getRotation function for getting the angles from the Quaternion representation.

APPENDIX D

SOFTWARE ARCHITECTURE OF PKD NEURAL NETWORK MAPPING

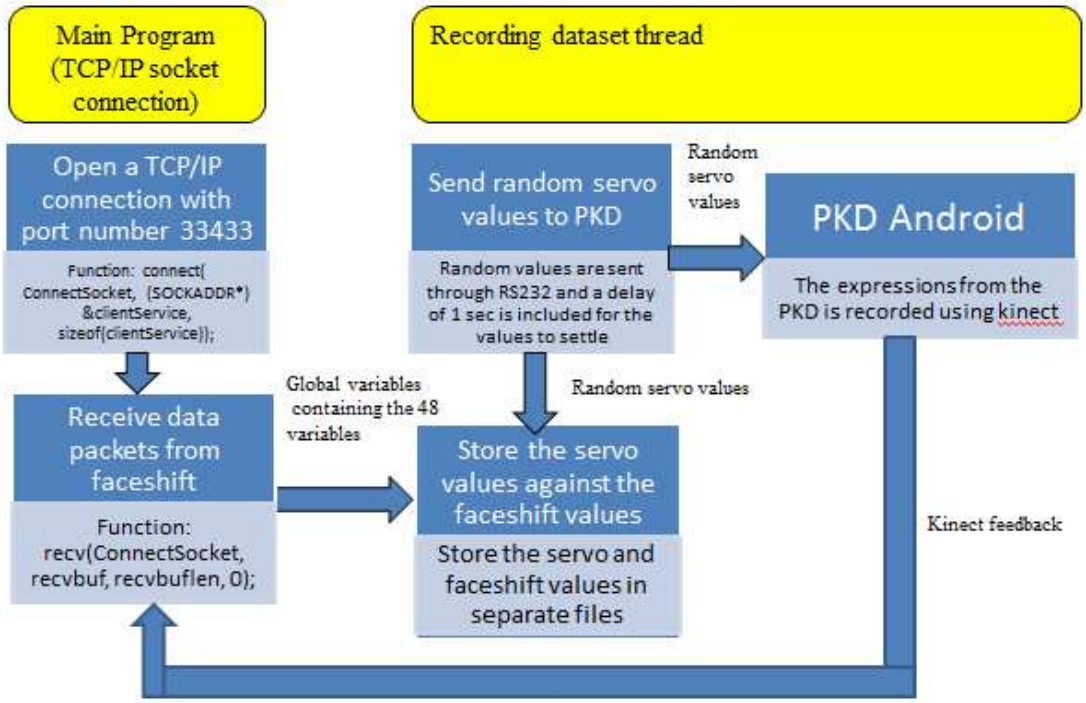


Figure D.1 Software Implementation of data collection

The figure D.1 shows the block diagram for collecting dataset for Neural Networks. The main program opens a TCP/IP socket for receiving the 48 blendshape variables from the faceshift software. The following code shows how it is done.

```
WSADATA wsaData;
WORD wVersionRequested;

wVersionRequested = MAKEWORD(2,2);
err = WSAStartup(wVersionRequested, &wsaData);
if (err != 0) {
	_endthread();
}
if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2) {
	WSACleanup();
	_endthread();
}

// Create a SOCKET for connecting to server
ConnectSocket = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if (ConnectSocket == INVALID_SOCKET) {
```

```
WSACleanup();
_endthread();
}

	// The sockaddr_in structure specifies the address family,
	// IP address, and port of the server to be connected to.
clientService.sin_family = AF_INET;
clientService.sin_addr.s_addr = inet_addr( "127.0.0.1" );
clientService.sin_port = htons( 33433 );

iResult = connect( ConnectSocket, (SOCKADDR*) &clientService,
sizeof(clientService));
if ( iResult == SOCKET_ERROR) {
closesocket (ConnectSocket);
WSACleanup();
_endthread();
}

if (ConnectSocket == INVALID_SOCKET) {
	closesocket(ConnectSocket);
	WSACleanup();
}
```

The connect function is responsible for establishing the socket connection. The code below shows how to extract the facial features from the data transmitted by TCP/IP

```
do {
		iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
		if ( iResult > 0 ){
			parserIn.received(iResult, recvbuf);
			while(msg=parserIn.get_message())
			{
				if(dynamic_cast<fs::fsMsgTrackingState*>(msg.get()))
				{
					fs::fsMsgTrackingState *ts =
dynamic_cast<fs::fsMsgTrackingState*>(msg.get());
					const fs::fsTrackingData &data = ts-
>tracking_data();


				}
			}
		}
		if(!parserIn.valid()) {
			printf("parser in invalid state\n");
			parserIn.clear();
		}
```

```
		// press keyboard c to call calibrate neutral
		int c = GetKeyState('C');
		if( c&0xF0 )
		{
			printf("Call calibrate neutral\n");
			fs::fsMsgCalibrateNeutral msg;
			std::string dataToSend;
			parserOut.encode_message(dataToSend, msg);
			send(ConnectSocket,dataToSend.data(),dataToSend.size(),0);
		}
	} while( iResult > 0 );
	_endthread();
```

The data.m.coefficients contains the 48 face shift variables. Another thread that is running in parallel will transmit random servo motors to the PKD. It then waits for the servo values to settle down and the 48 blendshape variables transmitted by the main function would be recorded against the servo values and is stored in separate files.

REFERENCES

[1] K. Tindell, H. Hansson and A. J. Wellings. Analysing real-time communications: Controller area network (CAN). Presented at Real-Time Systems Symposium, 1994., Proceedings. 1994.

[2] D. Jansen and H. Buttner. Real-time ethernet: The EtherCAT solution. *Computing and Control Engineering 15(1),* pp. 16-21. 2004.

[3] N. Villaroman, D. Rowe and B. Swan. Teaching natural user interaction using OpenNI and the microsoft kinect sensor. Presented at Proceedings of the 2011 Conference on Information Technology Education. 2011.

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng. ROS: An open-source robot operating system. Presented at ICRA Workshop on Open Source Software. 2009 .

[5] G. Husi and P. Szemes. Control of NI SbRIO-9631 prototype robot on the basis of hand movement. Presented at Robot Control. 2012.

[6] C. Rice. Prevalence of autism spectrum disorders: Autism and developmental disabilities monitoring network, united states, 2006. morbidity and mortality weekly report. surveillance summaries. volume 58, number SS-10. *Centers for Disease Control and Prevention* 2009.

[7] L. M. Oberman, E. M. Hubbard, J. P. McCleery, E. L. Altschuler, V. S. Ramachandran and J. A. Pineda. EEG evidence for mirror neuron dysfunction in autism spectrum disorders. *Cognitive Brain Research 24(2),* pp. 190-198. 2005.

[8] M. A. Gernsbacher, J. L. Stevenson, S. Khandakar and H. H. Goldsmith. Why does joint attention look atypical in autism? *Child Development Perspectives 2(1),* pp. 38-45. 2008.

[9] Y. Bar-Haim and O. Bart. Motor function and social participation in kindergarten children. *Social Development 15(2),* pp. 296-310. 2006.

[10] J. P. Piek, G. S. Bradbury, S. C. Elsley and L. Tate. Motor coordination and Social–Emotional behaviour in Preschool-aged children. *International Journal of Disability, Development and Education 55(2),* pp. 143-151. 2008.

[11] C. Breazeal and B. Scassellati. f 4 challenges in building robots that imitate people. *Imitation in Animals and Artifacts* pp. 363. 2002.

[12] E. Schopler, M. Lansing, R. Reichler and M. LM. PEP-3. *Psychoeducational Profile.Teach Individualized Psychoeducational Assessment for Children with Autism Spectrum Disorders.Austin: Pro-Ed* 2005.

[13] C. Min and A. H. Tewfik. Novel pattern detection in children with autism spectrum disorder using iterative subspace identification. Presented at Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. 2010.

[14] D. J. Ricks and M. B. Colton. Trends and considerations in robot-assisted autism therapy. Presented at Robotics and Automation (ICRA), 2010 IEEE International Conference on. 2010.

[15] B. Robins, K. Dautenhahn and J. Dubowski. Does appearance matter in the interaction of children with autism with a humanoid robot? *Interaction Studies 7(3),* pp. 509-542. 2006.

[16] D. Xu, J. Gilkerson, J. Richards, U. Yapanel and S. Gray. Child vocalization composition as discriminant information for automatic autism detection. Presented at Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE. 2009 .

[17] B. Scassellati, H. Admoni and M. Mataric. Robots for use in autism research. *Annu. Rev. Biomed. Eng. 14*pp. 275-294. 2012.

[18] J. Goetz, S. Kiesler and A. Powers. Matching robot appearance and behavior to tasks to improve human-robot cooperation. Presented at Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. the 12th IEEE International Workshop on. 2003 .

[19] S. Woods, K. Dautenhahn and J. Schulz. The design space of robots: Investigating children's views. Presented at Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on. 2004.

[20] M. Mori, K. F. MacDorman and N. Kageki. The uncanny valley [from the field]. *Robotics & Automation Magazine, IEEE 19(2),* pp. 98-100. 2012.

[21] N. J. Rinehart, J. L. Bradshaw, A. V. Brereton and B. J. Tonge. Movement preparation in high-functioning autism and asperger disorder: A serial choice reaction time task involving motor reprogramming. *J. Autism Dev. Disord. 31(1),* pp. 79-88. 2001.

[22] J. Fasola and M. J. Mataric. Using socially assistive human–robot interaction to motivate physical exercise for older adults. *Proc IEEE 100(8),* pp. 2512-2526. 2012.

[23] K. F. MacDorman and H. Ishiguro. Toward social mechanisms of android science. *Interaction Studies 7(2),* pp. 289-296. 2006.

[24] C. Bartneck, T. Kanda, H. Ishiguro and N. Hagita. Is the uncanny valley an uncanny cliff? Presented at Robot and Human Interactive Communication, 2007. RO-MAN 2007. the 16th IEEE International Symposium on. 2007.

[25] M. Shimada, T. Minato and S. Itakura. Uncanny valley of androids and its lateral inhibition hypothesis. Presented at Robot and Human Interactive Communication, 2007. RO-MAN 2007. the 16th IEEE International Symposium on. 2007 .

[26] S. Woods, K. Dautenhahn and J. Schulz. The design space of robots: Investigating children's views. Presented at Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on. 2004 .

[27] C. Breazeal and B. Scassellati. f 4 challenges in building robots that imitate people. *Imitation in Animals and Artifacts* pp. 363. 2002.

[28] B. Scassellati. Investigating models of social development using a humanoid robot. Presented at Neural Networks, 2003. Proceedings of the International Joint Conference on. 2003 .

[29] S. Gurbuz, T. Shimizu and G. Cheng. Real-time stereo facial feature tracking: Mimicking human mouth movement on a humanoid robot head. Presented at Humanoid Robots, 2005 5th IEEE-RAS International Conference on. 2005.

[30] C. Breazeal, A. Edsinger, P. Fitzpatrick and B. Scassellati. Active vision for sociable robots. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 31(5),* pp. 443-453. 2001.

[31] G. Cannata, M. D'Andrea and M. Maggiali. Design of a humanoid robot eye: Models and experiments. Presented at Humanoid Robots, 2006 6th IEEE-RAS International Conference on. 2006.

[32] G. Cannata and M. Maggiali. Implementation of listing's law for a tendon driven robot eye. Presented at Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on. 2006.

[33] J. Rajruangrabin and D. O. Popa. Robot head motion control with an emphasis on realism of neck–eye coordination during object tracking. *Journal of Intelligent & Robotic Systems 63(2),* pp. 163-190. 2011.

[34] S. K. Singh, S. D. Pieper, D. Popa and J. Guinness. Control and coordination of head, eyes and facial expressions of virtual actors in virtual environments. Presented at Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on. 1993.

[35] N. S. Pollard, J. K. Hodgins, M. J. Riley and C. G. Atkeson. Adapting human motion for the control of a humanoid robot. Presented at Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. 2002.

[36] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. Presented at Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. 2003.

[37] M. Vukobratović and B. Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid Robotics 1(01),* pp. 157-173. 2004.

[38] A. Safonova, N. Pollard and J. K. Hodgins. Optimizing human motion for the control of a humanoid robot. *Proc.Applied Mathematics and Applications of Mathematics 78*2003.

[39] M. Ruchanurucks, S. Nakaoka, S. Kudoh and K. Ikeuchi. Humanoid robot motion generation with sequential physical constraints. Presented at Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. 2006.

[40] M. J. Mataric. Getting humanoids to move and imitate. *Intelligent Systems and their Applications, IEEE 15(4),* pp. 18-24. 2000.

[41] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. Presented at Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction. 2007.

[42] T. Asfour, P. Azad, F. Gyarfas and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics 5(02),* pp. 183-202. 2008.

[43] T. Chen and R. R. Rao. Audio-visual integration in multimodal communication. *Proc IEEE 86(5),* pp. 837-852. 1998.

[44] H. McGurk and J. MacDonald. Hearing lips and seeing voices. 1976.

[45] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. Presented at Image Processing. 2002. Proceedings. 2002 International Conference on. 2002.

[46] Macedo, Marcio Cerqueira de Farias, A. L. Apolinario and Souza, Antonio Carlos dos Santos. A robust real-time face tracking using head pose estimation for a markerless AR system. Presented at Virtual and Augmented Reality (SVR), 2013 XV Symposium on. 2013.

[47] F. Wilbers, C. Ishi and H. Ishiguro. A blendshape model for mapping facial motions to an android. Presented at Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. 2007.

[48] T. Wu, N. J. Butko, P. Ruvulo, M. S. Bartlett and J. R. Movellan. Learning to make facial expressions. Presented at Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on. 2009.

[49] I. Ranatunga, M. Beltran, N. A. Torres, N. Bugnariu, R. M. Patterson, C. Garver and D. O. Popa. "Human-robot upper body gesture imitation analysis for autism spectrum disorders," in *Social Robotics*Anonymous 2013.

[50] "http://sine.ni.com/nips/cds/view/p/lang/en/nid/210420," .

[51] "http://www.pololu.com/product/1356," .

[52] "http://www.darpa.mil/NewsEvents/Releases/2013/07/11.aspx," .

[53] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. Presented at Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. 2004.

[54] "http://gazebosim.org/wiki/DRC," .

[55] L. A. Schwarz, A. Mkhitaryan, D. Mateus and N. Navab. Human skeleton tracking from depth data using geodesic distances and optical flow. *Image Vision Comput. 30(3),* pp. 217-226. 2012.

[56] B. Thuilot, P. Martinet, L. Cordesses and J. Gallice. Position based visual servoing: Keeping the object in the field of vision. Presented at Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. 2002.

[57] G. L. Mariottini, G. Oriolo and D. Prattichizzo. Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. *Robotics, IEEE Transactions on 23(1),* pp. 87-100. 2007.

[58] "http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html,".

[59] M. Inaba, F. Kanehiro, S. Kagami and H. Inoue. Two-armed bipedal robot that can walk, roll over and stand up. Presented at Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on. 1995 .

[60] F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, S. Kajita, K. Yokoi, H. Hirukawa, K. Akachi and T. Isozumi. The first humanoid robot that has the same size as a human and that can lie down and get up. Presented at Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. 2003 .

[61] (). . Available: http://www.humanoids2013.com/.

[62] (). *Global Future 2045 International Congress*. Available: http://gf2045.com/.

BIOGRAPHICAL INFORMATION

Suresh Sampathkumar was born in Bangalore, India. He gained his Bachelors of Engineering degree in Electronics and Communications from Dr. Ambedkar Institute of Technology, Bangalore in 2005 and gained his Master of Science degree in Electrical Engineering at University of Texas at Arlington in 2013. His main research interests are in Robotics with real-time implementation in an Embedded System platform. He is also interested in Digital Signal Processing and Computer Vision